DATAMOST

Computer persönlich

Einführungskurs:

COMMODORE 64

William B. Sanders

Eine praxisnahe Anleitung für die Bedienung.
Programmieren in Basic. Mit Beispielen für:
Grafik, Dateien, Töne.

Einführungskurs: Commodore 64

William B. Sanders

Einführungskurs: Commodore 64

Eine praxisnahe Anleitung für die Bedienung. Programmieren in Basic. Mit Beispielen für: Grafik, Dateien, Töne

Deutsche Übersetzung:
Peter Wassmann

Markt & Technik Verlag

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Sanders, William B.:
Einführungskurs: Commodore 64 : e. prakt. Anleitung für d. Bedienung ; programmieren in Basic ; mit Beispielen für Grafik, Dateien, Töne / William B. Sanders.
Haar bei München : Markt-und-Technik-Verlag, 1984.
(Computer persönlich)
Einheitssacht.: The elementary Commodore 64 <dt.>
ISBN 3-89090-017-8

Die Informationen im vorliegenden Buch werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen. Trotzdem können Fehler nicht vollständig ausgeschlossen werden. Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien. Die gewerbliche Nutzung der in diesem Buch gezeigten Modelle und Arbeiten ist nicht zulässig.

»Commodore 64« ist eine Produktbezeichnung der Commodore Büromaschinen GmbH, Frankfurt, die ebenso wie der Name »Commodore« Schutzrecht genießt. Der Gebrauch bzw. die Verwendung bedarf der Erlaubnis der Schutzrechtsinhaberin.

Titel der amerikanischen Originalausgabe
»THE ELEMENTARY COMMODORE 64«
by William B. Sanders
ISBN 0-88190-001-X

English edition © Copyright 1983
by DATAMOST, Inc., Chatsworth, CA 91311-2750, USA
All rights reserved

ISBN 3-89090-017-8

Übersetzung © 1984 by Markt & Technik, 8013 Haar bei München
Alle Rechte vorbehalten
Einbandgestaltung: Grafikdesign Heinz Rauner
Zeichnungen: René Nestler
Druck: Eimannsberger, München
Printed in Germany

Inhaltsverzeichnis

Vorwort 9

1 Einführung 11

1.1 Aufstellen des COMMODORE-64 und der peripheren Geräte 17
1.2 Betriebsfertig 24
1.3 Die COMMODORE-64 Tastatur 30
1.4 Zusammenfassung 35

2 Grundlagen 37

2.1 Print 40
2.2 Ihr erstes Programm 41
2.3 Erstellen eines Programms 43
2.4 Verwendung des Editors 49
2.5 Grundlegende mathematische Operationen 55
2.6 Zusammenfassung 59

3 Wir machen weiter 61

3.1 Variablen 63
3.2 Input und Output (I/O) 74
3.3 Schleifen mit FOR/NEXT 80
3.4 Zusammenfassung 87

4 Verzweigungen im Programm 89

4.1 Verzweigen 92
4.2 Vergleichsoperatoren 95
4.3 AND/OR/NOT 98
4.4 Unterroutinen 101
4.5 Tabellen 107
4.6 Zusammenfassung 114

5 Organisieren von Programmteilen **115**

5.1 Formatieren von Text 117
5.2 Aufteilen von Strings 120
5.3 Organisieren der Eingabedaten 132
5.4 Organisieren der Datenverarbeitung 133
5.5 Organisation der Ausgabe 135
5.6 Rollen des Bildschirms 137
5.7 Zusammenfassung 139

6 Fortgeschrittene Anwendungen **141**

6.1 Der ASCCI-Code und die CHR$-Funktion 143
6.2 POKE und PEEK 147
6.3 POKEn im Bildschirmspeicher 151
6.4 Eine ganze Menge Töne 155
6.5 Zusammenfassung 159

7 Benutzen der Grafik **161**

7.1 Bildschirmgrafik 164
7.2 Bringen Sie Farbe in Ihre Grafik 167
7.3 Spritegrafik 181
7.4 Zusammenfassung 192

8 Dateiverarbeitung **195**

8.1 Datendateien auf Band 197
8.2 OPEN, INPUT#, PRINT# und CLOSE 199
8.3 Sequentielle Daten auf Diskette 207
8.4 Zusammenfassung 213

9 Ihr Drucker **215**

9.1 Die Ausgabe von Text auf dem Drucker 219
9.2 CHR$-Funktionen 222
9.3 Das Ausgeben von Grafik 230
9.4 Zusammenfassung 235

10 Programmhilfen und Hinweise **237**

10.1 COMMODORE-64 User Groups 239
10.2 COMMODORE-64 Zeitschriften 240
10.3 Der COMMODORE-64 spricht mehrere Sprachen 242
10.4 Sortier-Routinen 245
10.5 Tastaturtricks 249
10.6 Hilfsprogramme 252
10.7 Textverarbeitung 252
10.8 Datenbankprogramme 255
10.9 Programme für kommerziellen Einsatz 255
10.10 Grafikpakete 257
10.11 Hardware 257
10.12 Modems und Datenkommunikation 258
10.13 Zusammenfassung 258

Anhang A: C-64 WEDGE **261**

Anhang B: COMMODORE-64 Kommandoübersicht **263**

Anhang C: Zeichencodes **270**

Index **273**

Vorwort

Meine erste formelle Einweisung in die Arbeit mit einem Computer erhielt ich 1966. Damals erzählte unser erfahrener Lehrer, wir hätten ausgesorgt, wenn wir den untersten Level in der Handhabung eines Computers erreicht hätten. Als Ergebnis dieser Philosophie konnten wir binär zählen und Werte oktal und dann dezimal konvertieren. Wir schrieben Codes für die Computer, wobei wir nur "0" und "1" benutzten. Wir lernten aber auch, Programme in einer höheren Programmiersprache, FORTRAN, zu schreiben. Verglichen mit dem Binär- oder Oktalcode war FORTRAN leicht. Das Problem war, daß wir nie an einem Bildschirm arbeiten konnten, und obwohl wir ein sehr gutes theoretisches Verständnis der Abläufe im Computer hatten, lernten wir nicht sehr viel über die Zusammenarbeit mit dem Computer. Wir benutzten Lochkarten, sandten unsere Programme über Telefonleitungen in einen Computer in der Universität und warteten dann eine Woche, um die Ergebnisse zu erhalten. War im Programm ein Fehler, mußten wir das gesamte Programm für einen weiteren Versuch nochmals übersenden.

Seit dieser Zeit haben sich die Computer wie auch die Benutzer gewandelt. Um den Umgang mit einem Computer zu erlernen, ist es nicht mehr notwendig zu wissen, wie dieser intern arbeitet, oder die Theorie zu erlernen, die hinter den Operationen steht. Es ist richtig, daß man mehr mit dem Computer machen kann, wenn man ein detailliertes Verständnis der Theorie und der Operationen eines Computers hat. Dies ist jedoch nicht mehr erforderlich. Man kann programmieren lernen und sich erst zu einem späteren Zeitpunkt mehr technische Einzelheiten über die Arbeitsweise des Computers aneignen. Die meisten Leute lernen ja auch das Autofahren, ohne Kenntnisse über die kompliziertenn Abläufe im Verbrennungsmotor ihres Autos zu besitzen.

Eine weitere große Veränderung trat durch den Übergang von Großcomputern mit Bildschirmen zu kleinen Schreibtischcomputern ein. Sie sind vollwertige Computer. Deshalb sind wir nicht mehr auf einen Großcomputer angewiesen und können alles selbst erledigen. Daraus ergibt sich, daß Sie sich nicht mehr an einen vorgegebenen Dienstplan und an Bestimmungen halten müssen, um Rechenzeit zu erhalten oder für "Computerzeit" zu bezahlen haben. Sie erstellen Ihren eigenen Plan und sind Ihr eigener Chef. Aus diesem Grund ist es nicht notwendig, lange über die organisatorischen Aspekte der Hauptspeicherzeitvergabe zu diskutieren oder über alle weite-

ren Probleme, die auftreten, wenn mehrere Personen "gleichzeitig" den Computer benutzen. Wir wollen somit gleich zum Kern der Sache kommen, nämlich, wie Sie Ihren Computer programmieren können.

Der Zweck dieses Buches liegt hauptsächlich darin, Sie im Umgang mit Ihrem Computer und in der Programmiersprache BASIC zu unterweisen. Es vermittelt ein Grundwissen. Erwarten Sie deshalb bei der Arbeit mit dem Buch nicht, alles über Ihren Commodore-64 zu lernen. Wenn Sie mit diesem Buch zu Ende sind, werden Sie erst sehen, wieviel mehr Sie mit Ihrem Computer anfangen können. Je mehr Sie lernen, desto größer wird das noch zu erforschende Gebiet. Wenn Sie jedoch die folgenden Anweisungen nachvollziehen und die kleinen Beispiele eintippen, werden Sie schnell lernen, wie Sie mit den BASIC-Befehlen Ihres Commodore-64 Programme schreiben können.

Erwarten Sie aber auch nicht, alles auf einmal zu lernen. Seien Sie mit sich und Ihrem Computer nachsichtig, und Sie werden überrrascht sein, wieviel Sie lernen. Wenn Sie ein Kommando oder einen Vorgang nicht sofort verstehen, können Sie jederzeit später wieder darauf zurückkommen. Experimentieren und spielen Sie mit Ihren Programmen. Denken Sie sich verschiedene Anwendungen für Ihren Computer aus, und versuchen Sie dann ein Programm dafür zu schreiben. Haben Sie vor allem keine Angst, einen Versuch zu starten. Mit jedem Versuch und jeder weiteren Zeile werden Sie Fortschritte machen. Auch falls es mal langsam vorangeht, wird das angesammelte Wissen zum Verständnis führen.

1

Einführung

Dieses Buch wurde als Hilfe zur Handhabung Ihres neuen COMMODORE-64 und als Anleitung zum Programmieren geschrieben. Es ist nicht für fortgeschrittene Programmierer oder erweiterte Anwendungen gedacht. Es ist sozusagen der erste Schritt für Anfänger auf dem COMMODORE-64 Computer. Das gesamte Buch ist in Form von Anleitungen geschrieben; wenn Sie es durchgearbeitet haben, werden Sie in der Lage sein, Programme zu schreiben und zu benutzen.

Für eine systematische Einarbeitung in dieses Buch ist es empfehlenswert, ganz zu Beginn zu starten und es Schritt für Schritt durchzuarbeiten. Ich habe versucht, die Kapitel logisch aufeinander aufzubauen. Aus dem Versuch, Sprünge zu machen, kann resultieren, daß Sie einige wichtige Computeroperationen nicht verstehen. Die einzige Ausnahme von dieser Regel ist das letzte Kapitel, in dem sich eine Anzahl von Vorschlägen für Programme befinden, die Ihnen beim Programmieren helfen sollen (sog. Dienstprogramme). Ebenso befinden sich dort Programmbeschreibungen für Geschäftsprogramme, Textverarbeitung usw. Wenn Sie dieses Kapitel durchgearbeitet haben, wäre es gut, wenn Sie einen Blick auf jene aufgeführten Programme werfen, um während des Arbeitens mit diesem Buch schon zu wissen, ob eines davon Ihren Bedürfnissen entspricht. Sie brauchen keines davon zu kaufen. Sie werden jedoch einige davon sehr nützlich finden.

Das erste, was Sie über Ihren Computer wissen sollten, ist, daß er Sie nicht "beißt", jedoch einer gewissen Sorgfalt bedarf. Sie können Disketten, Bänder und Informationen zerstören. Wenn Sie aber einige einfache Regeln einhalten, sind Sie auf dem richtigen Weg. Jeder von uns benutzt täglich hochentwickelte elektronische Geräte, wie Stereoanlage, Fernseher oder Videorecorder - und auch dort ist eine gewisse Sorgfalt erforderlich. Ansonsten brauchen Sie sich nicht zu fürchten. Da Ihr Computer ebenfalls ein elektronisches Gerät ist, werden Sie ihn zerstören, wenn Sie Wasser oder andere Flüssigkeiten über ihn gießen. Seien Sie sehr vorsichtig, aber auch unternehmenslustig und beginnen Sie. Erinnern Sie sich, daß es praktisch unmöglich ist, ein Programm zu schreiben, das Ihr Gerät oder die elektronischen Schaltkreise in der Maschine zerstört. Schlimmstenfalls kann eines der von Ihnen erstellten Programme die Informationen auf der Diskette oder auf dem Band löschen. Überall in diesem Buch gebe ich Tips, wie man die Dinge richtig und falsch macht. Behandeln Sie also Ihren Computer genau wie Ihren Microwellenherd, Ihr Radio oder ein anderes elektronisches Gerät - mit Vorsicht, aber ohne Angst.

Im Augenblick ist es noch nicht nötig, alle Begriffe aus dem Computer-Jargon zu lernen; aber einige Ausdrücke sind für ein besseres Verständnis der Arbeitsweise des Computers erforderlich. Später werden weitere neue Ausdrücke erläutert, doch grundsätzlich ist das Buch in gut verständlichem Deutsch verfaßt. Nichtsdestoweniger sollten Sie vor dem eigentlichen Start folgende Begriffe kennen.

Hardware

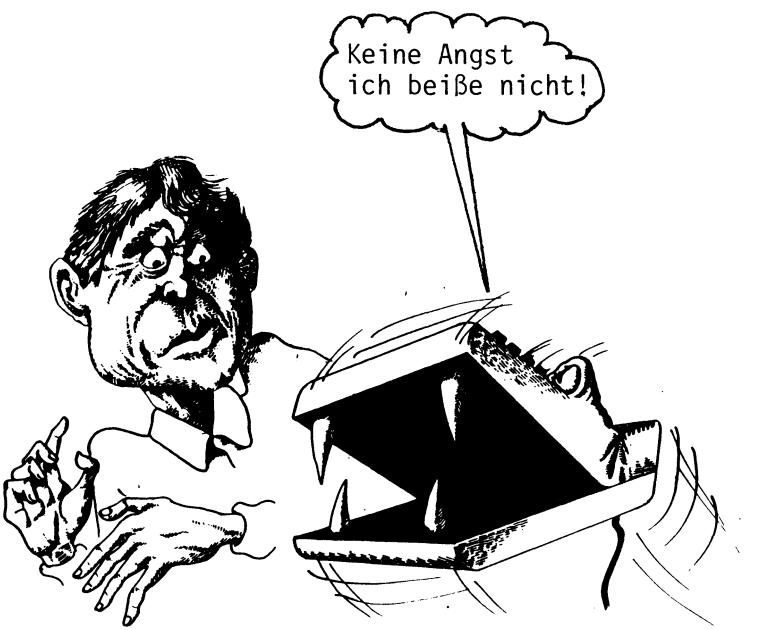
Der Begriff Hardware gilt für Ihre Maschine und alle elektronischen Teile. Alles, angefangen bei der Tastatur über die Schaltkreise zu den kleinen schwarzen Chips in Ihrem Computer, wird "Hardware" genannt. Sie werden auch den Ausdruck "Firmware" hören. Dies ist ein anderer Typ der Hardware, auf dem Programme geschrieben werden. Die Chips, genannt "Prom" oder "Eprom", speichern Informationen genau wie Diskettten oder Bänder. Firmware ist entweder in Ihrem Computer, in einem Extragehäuse oder auf Karten, die von hinten in Ihren COMMODORE-64 gesteckt werden. Ein biologischer Vergleich zu diesem Thema: Der Körper, vor allem das Gehirn, ist die Hardware und Firmware ist wie angeborene oder erworbene Intelligenz.

Software

Software ist das Programm, das Ihrem Computer mitteilt, wie er zu arbeiten hat. Alle Informationen, die vom Speicher des Computers aufgenommen werden, bedeuten "Software". Sie steht analog zu Ihren Gedanken oder Ideen. Sehen Sie die Hardware als Gehirn des Computers an und die Software als die einzuspeichernden Ideen. Software ist für den Computer das gleiche wie Schallplatten für Ihren Plattenspieler. Software wird entweder im "Random Access Memory" (ein Speicher, den man sowohl lesen als auch beschreiben kann, Abk: RAM) oder im "Read Only Memory" (Speicher, die man nur lesen kann, Abk: ROM) gespeichert. Firmware ist Hardware mit "eingebrannter" Software.

RAM - Sie hören vielleicht von Leuten, die Ihr RAM erweitert haben. Dies ist jener Teil des Computerspeichers, in den Sie Informationen in Form von Daten oder Programmen eingeben können. Je größer der Speicher ist, desto längere Programme und größere Datenmengen können Sie eingeben. Denken Sie bei RAM an ein Lagerhaus. Wenn Sie das erste Mal Ihren Computer einschalten, ist das Lagerhaus noch leer (eine Meldung sagt: 38911 BASIC BYTES FREE - 38911 Bytes freier Speicher); aber wenn Sie Programme laufen lassen und Informationen eingeben, beginnt sich das Lager zu füllen. Je größer es ist, desto mehr Informationen können Sie darin speichern; wenn es voll ist, müssen Sie aufhören. COMMODORE-64

Computer haben 64K RAM. das "K" steht für "Kilobytes" oder "tausend Bytes", genau gesagt: 1024 Bytes. Die neuen Plattenspeichersysteme werden in "Megabytes" oder "1 Million Bytes", genau: 1.024.000 Bytes, gemessen. Erwähnen Sie dies bei Ihrer nächsten Party, und Sie werden jeden beeindrucken. Sie müssen nur wissen, daß Bytes eine Maßeinheit darstellen, ähnlich wie Liter, Meter oder Zentimeter. Je mehr Bytes ein Computer besitzt, desto mehr Platz für Programme ist vorhanden.



ROM - Der zweite Typ von Computerspeichern ist der "Read Only Memory". Dieser Speichertyp ist ein fester Bestandteil des Computerchips. Die Programmiersprache des COMMODORE-64, BASIC, ist im ROM gespeichert. Den Unterschied zwischen RAM und ROM sehen Sie, wenn Sie den Computer ausschalten. Die Informationen im RAM verschwinden, während die im ROM erhalten bleiben. Lassen Sie sich aber nicht irritieren. Sie können alles, was sich im RAM befindet, auf Diskette oder Band speichern und sich dann wieder holen. Wie dies geschieht, werde ich später noch erklären.

Jetzt sind Sie mit einigen Begriffen genügend vertraut, um Ihren Computer nicht zu fürchten. Schalten Sie ihn ein und lassen Sie ihn arbeiten. Haben Sie Ihren Computer bereits aufgestellt und er arbeitet richtig, können Sie direkt zum nächsten Abschnitt "Betriebsfertig" springen.

1.1 Aufstellen des COMMODORE-64 und der peripheren Geräte

Nach Durcharbeiten dieses Abschnittes sollten Sie den COMMODORE-64 anschließen und einschalten. Vorher müssen wir aber noch einige Vorbereitungen treffen. Wenn Sie Ihrem Computer ohne Diskettenstation oder einen Datenrecorder gekauft haben, können Sie damit zwar arbeiten, aber Sie brauchen eine Commodore-Kassetteneinheit oder ein Diskettenlaufwerk, um Ihre eingegebenen Informationen abzuspeichern. Wenn Sie diese zusätzlichen Geräte nicht besitzen, können Sie direkt zum Abschnitt "Betriebsfertig" übergehen.

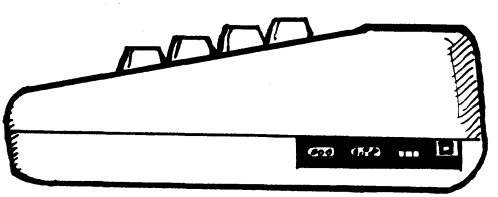
Datenrecorder

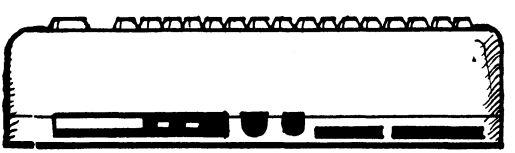
Wenn Sie einen Datenrecorder benutzen (mit oder ohne Diskettenbetriebssystem), ist es sehr einfach, ihn aufzustellen. An Ihrer Commodore-C2N-Kassetteneinheit befindet sich ein Kabel für die Verbindung mit dem Computer. Nehmen Sie das Kabel und stecken Sie es in den schmalen Anschluß, der sich hinten links an Ihrem Computer befindet (direkt hinter den Tasten "4","5" und "6"). Vergewissern Sie sich, daß das Kabel korrekt mit den Stiften des Anschlußes verbunden ist, stellen Sie aber auf keinen Fall die Ver-

bindung mit Gewalt her. Das ist alles. Ihr Kassettenrecorder ist nun betriebsbereit. Benutzen Sie normale Kassetten - gewöhnlich sind die 5-Minuten-Bänder die besten.

Diskettenlaufwerk

Zu Ihrem COMMODORE-64 sollten Sie das VIC-1541 Diskettenlaufwerk benutzen. (Das VIC-1540 wird nur dann korrekt mit Ihrem COMMODORE-64 arbeiten, wenn Sie ein zusätzliches Spezialchip einbauen.) Um Ihr Diskettenlaufwerk anzuschließen, befestigen Sie das eine Ende des Diskettenkabels am seriellen Anschluß hinten an Ihrem Computer (direkt rechts vom Kassettenanschluß, hinter den Tasten "7" und "8") und das andere Ende in der Fassung an der Rückseite Ihres Diskettenlaufwerks, direkt über der "Sicherung". Jetzt stecken Sie das Stromkabel Ihres Diskettenlaufwerks in die dreizackige Fassung zwischen dem "Ein/Aus"-Schalter und der Sicherung. Wenn alles andere verbunden ist, können Sie das Stromkabel des Laufwerks in die Steckdose führen und den Schalter auf "ON" drücken. Tun Sie dies aber jetzt noch nicht.





Fernseher oder Monitor

Damit Sie auch sehen, was in Ihrem Computer vorgeht, brauchen Sie einen Monitor. Für manche Computer ist es notwendig, einen RF-Modulator zu erwerben. Ihr COMMODORE-64 wird jedoch schon mit eingebautem RF-Modulator geliefert. Stecken Sie nun das eine Ende des Commodore-64-Verbindungskabels in die TV-Buchse an der Rückseite Ihres Computers, direkt hinter der "+"-Taste, und das andere Ende in die "Buchse" Ihres Fernsehers. Diese Buchse gehört zur Antennenverbindung und ist duch "VHF" auf der Rückseite Ihres Fernsehgerätes markiert.

Eine andere Möglichkeit ist ein Monitor anstelle des Fernsehers. Grundsätzlich erfüllt der Monitor den gleichen Zweck wie das Fernsehgerät; er hat jedoch eine höhere Auflösung und ist sehr nützlich, wenn Sie viel mit Textverarbeitung zu tun haben. Um eine Verbindung zum Monitor zu schaffen, brauchen Sie ein spezielles Kabel, das direkt neben dem seriellen Ausgang eingesteckt wird. Das fünfpolige Audiokabel, das Sie in Elektronikgeschäften erhalten, ist ein guter Kauf, da Sie es auch für Ihre Stereoanlage benutzen können.

Im folgenden beschreibe ich einige Monitore und Fernsehgeräte, die Sie für Ihren COMMODORE-64 benutzen können.

Verschiedene Fernsehgeräte

Fernsehgeräte gibt es in vielen Ausführungen und Größen, in Farbe oder Schwarzweiß. Seien Sie vorsichtig bei der Auswahl des Gerätes. Nicht jeder Fernseher arbeitet gut in Verbindung mit Ihrem COMMODORE-64. Als ich meinen Fernseher kaufte - ein Farbgerät wegen der Grafik - schaute ich mich zuerst einmal in Computerläden um, welche Geräte dort benutzt werden. Am günstigsten ist ein einfaches Schwarzweißgerät. Es hat eine bessere Auflösung als ein Farbgerät, ist nicht so teuer und eignet sich gut für Textverarbeitung. Das beste dabei ist, daß Sie es schon für weniger als 200 DM bekommen. Wie auch immer, vergewissern Sie sich zuerst, ob das Gerät zusammen mit Ihrem COMMODORE-64 funktioniert.

Verschiedene Monitore

Grüner Bildschirm - Dieses Gerät produziert grüne Schrift auf schwarzem Grund und kostet zwischen 200 und 500 DM. Die Grün-auf-Schwarz-Darstellung ist gut für Textverarbeitung und Programme ohne Grafikdarstellung und sehr angenehm für die Augen. Da auf dem Gerät nur die Farben Grün und Schwarz verwendet werden, ist es nicht für farbige Grafik geeignet.

Schwarzweiß - Dieser Bildschirm ist im wesentlichen dem grünen Bildschirm gleichzusetzen. Er ist teurer als ein Schwarzweißfernseher. Da er aber eine bessere Auflösung hat, ist der kleine Aufpreis gerechtfertigt. Wenn Sie überlegen, ob Sie einen Schwarzweiß-Monitor kaufen sollen, vergleichen Sie seine Auflösung mit der eines Schwarzweißfernsehers und prüfen Sie, ob sich die Extraausgabe nicht doch lohnt.

Farbe - Dieser Monitortyp ist der teuerste, aber für Personen, die sehr viel mit Farbgrafik arbeiten, unumgänglich. Er enthält die erforderliche Hochauflösung, um die Grafiken im Detail zu sehen. Manche Farbmonitore erfordern aber eine spezielle Schnittstelle. Prüfen Sie vor dem Kauf des Bildschirms, ob Sie für Ihren COMMODORE-64 eine Schnittstelle bekommen.

Drucker

Dieser Abschnitt informiert Sie über die verschiedenen Arten von Druckern und deren Aufstellung. Wenn Ihr Drucker bereits aufgestellt ist und richtig arbeitet, schauen Sie sich die Tips für den optimalen Gebrauch Ihres Druckers in Kapitel 9 an.

Druckertypen

Es gibt drei verschiedene Druckertypen.

MATRIXDRUCKER - Er ist der bekannteste Drucker auf dem Markt. Dieser Drucker hat eine Reihe von kleinen Nadeln, die heraus-

schießen, einen Punkt darstellen und so Text oder Grafik ausgeben. Der Vorteil der Matrixdrucker liegt in den niedrigen Kosten und der Tatsache, daß sie Text **und** Grafik ausgeben können. Matrixdrucker erreichen in der Betriebsart "Schönschreiben" fast Korrespondenzqualität und bieten gewöhnlich mehrere Schriftarten. In Kapitel 9 zeigen wir Beispiele von unterschiedlichen Druckarten der Matrixdrucker. Wir benutzen für die meisten Beispiele den VIC-1515-Drucker von Commodore, da dieser kompatibel mit dem COMMODORE-64 ist. (Parallele Schnittstellen für Matrixdrucker sind lieferbar. Siehe Korrespondenzqualität.)

KORRESPONDENZQUALITÄT - Für Anwender, die ihren Computer hauptsächlich für Textverarbeitung benutzen, sind Drucker mit Korrespondenzqualität gut geeignet. Die meisten sind Schönschreibdrukker mit Typenrad und arbeiten genau wie eine Schreibmaschine. Diese Drucker sind nicht für Grafikausgaben geeignet, jedoch für die Anwender am besten, die Wert auf ein perfektes Schriftbild für Manuskripte oder Dokumente legen. Sie sind aber relativ teuer. Zudem genügt für die meisten Anwendungen ein Matrixdrucker. Bevor Sie sich entscheiden, sollten Sie erst einmal vergleichen. Da Sie zum Anschluß eines Schönschreibdruckers an Ihren COMMODORE-64 eine spezielle Schnittstelle benötigen, lassen Sie sich den Drucker, angeschlossen mit dieser Schnittstelle, vor dem Kauf vorführen. Da der COMMODORE-64 einen seriellen Ausgang (anstatt eines parallelen Ausgangs) hat, werden Sie wahrscheinlich einen Drucker bekommen, der seriell kompatibel ist.

THERMODRUCKER - Wenn Sie nicht so viel Geld ausgeben möchten, ist vielleicht der Thermodrucker der richtige für Sie. Dieser Drucker arbeitet mit Spezialpapier, gewöhnlich Endlospapier und fertigt ein "Bild" Ihres Monitors an. Er kann sowohl Text als auch Grafik ausgeben. Die Qualität der Ausgabe ist jedoch relativ gering und das Spezialpapier sehr teuer. Der Vorteil liegt im leichten Gewicht und den kleinen Abmessungen. Diese Drucker sind daher sehr geeignet für Leute, die mit dem Computer verreisen und schnell mal eine Kopie brauchen. Achten Sie, genau wie beim Matrix- und Schönschreibdrucker, vor dem Kauf darauf, daß die Schnittstelle zu Ihrem COMMODORE-64 paßt.

Ein Ratschlag

Überlegen Sie vor dem Kauf eines Drucker den Verwendungszweck und die Besonderheiten der verschiedenen Druckerarten! Lassen Sie sich nicht von einem Verkäufer überzeugen, daß ein bestimmter Drucker an den COMMODORE-64 anzuschließen ist, wenn Sie es nicht gesehen haben. Selbst ein Verkäufer mit den besten Absichten (er glaubt z.B., daß jener Drucker für Ihre Bedürfnisse der beste sei) weiß vielleicht nicht, daß der Drucker an Ihren Computer nicht anschließbar ist. Nur eine Demonstration kann alle Zweifel beseitigen!

Wenn Sie einen VIC-1515 oder einen ähnlichen Commodore-Drucker kaufen, ist es sehr einfach, ihn anzuschließen. Haben Sie ein Diskettenlaufwerk, stecken Sie das eine Ende des Kabels in den freien Anschluß an der Rückseite des Diskettenlaufwerks und das andere in den Drucker. Dies nennt man "Zwischenschleifen" des Druckers. Wenn Sie einen Datenrecorder haben, stecken Sie das Kabel neben dem Kassettenanschluß an der Rückseite des Computers ein. Es ist der gleiche Anschluß, in den Sie das Diskettenlaufwerk stecken würden. An der Rückseite des Druckers befindet sich ein Schalter, den man auf "T", "4" oder "5" stellen kann. Stellen Sie ihn auf "4". (Die "T"-Position ist für "Test" und die Position "5" für die Identifizierung des Gerätes als "Gerät 5". Wir benutzen in unseren Beispielen den Drucker als "Gerät 4", stellen Sie ihn deshalb bitte auch so ein.)

Warnung

Stecken oder entfernen Sie NIEMALS ein Kabel in die/aus der Schnittstelle während das Gerät EINGESCHALTET ist! Auch wenn Sie genug Geld übrig haben, sich jedesmal ein neues Chip zu kaufen (da Sie Ihr ursprüngliches bei solch einer Behandlung zerstören könnten), wäre das doch sehr ärgerlich.

Weitere Geräte

Benutzer, die außer ihrem Diskettenlaufwerks, dem Fernseher oder Monitor und dem Drucker im Moment nichts anschließen, können direkt zum nächsten Abschnitt übergehen. Falls Sie planen, Ihren COMMODORE-64 zu erweitern oder weitere Geräte anzuschließen, sollten Sie zuerst folgende Absätze lesen.

Mehrere Ausgänge

Die besten Eigenschaften des Commodore-64 sind seine Ausbau- und Anpassungsfähigkeit. Die verschiedenen Ausgänge und Anschlüsse an Ihrem Computer können Sie dazu benutzen, um verschiedene Geräte anzuhängen und somit Ihr System zu erweitern.

MODEM - Ein Modem ist ein Gerät, das ihrem Computer die Möglichkeit gibt, mit anderen Computern über die Telefonleitung zu kommunizieren. Diese Geräte erfordern, daß Sie Ihr Telefon mit einem Teil des Modems verbinden oder den Hörer in einen Akkustikkoppler legen. Das VICMODEM kann mit dem COMMODORE-64 zusammen benutzt werden, indem Sie es einfach in den "User Port", rechts neben dem Kasettenausgang stecken und es mit der Telefonleitung verbinden. In Amerika können Sie sich mit dem Modem auch in bestehende Informationszentren (Source) einwählen, um alles, angefangen vom Wetterbericht bis hin zum Flugticket, zu erhalten. Bevor Sie bei uns ein Modem anschließen, erkundigen Sie sich bei der Post, ob dieses auch zugelassen ist.

Z-80 CP/M - Dieses Bauteil wird direkt in den Gehäuseanschluß eingesteckt, um Ihre Maschine in einen Z-80-Computer zu verwandeln. Es ermöglicht Ihnen, die umfassenden CP/M-Programme laufen zu lassen. Unter den 2000 CP/M-Programmen auf dem Markt sind nur wenige, die dann nicht laufen.

NOCH MEHR ERWEITERUNGEN - Es gibt noch eine Reihe zusätzlicher Erweiterungen um Ihren COMMODORE-64 zu einer vielseitigen Maschine zu machen. Spezielle Schnittstellen erlauben Ihnen den Anschluß einer Vielzahl von peripheren Geräten, wie z.B. verschiedenen Diskettenlaufwerken, Druckern und weiteren Geräten für andere Computer. Nun ist der COMMODORE-64 an sich schon ein her-

vorragender Computer, aber er ist auch voll ausbaufähig und wird dadurch noch besser.

1.2 Betriebsfertig

System-Check

Nachdem Sie alle benötigten Geräte an Ihren COMMODORE-64 angesteckt haben, schalten Sie ihn, zusammen mit Ihrem Fernseher oder Monitor, Diskettenlaufwerk und Drucker, ein. Der Schalter befindet sich rechts, direkt neben dem Stromkabelanschluß. Bringen Sie ihn in die ON-Position und schalten Sie Ihren Fernseher oder Monitor ein. Wenn alles angeschlossen ist, bekommen Sie auf dem Bildschirm folgende Meldung:

```
**** COMMODORE 64 BASIC V2 ***

64K RAM SYSTEM 38911 BASIC BYTES FREE

READY.
```

Wenn Sie einen Farbfernseher haben, erscheinen die Buchstaben hellblau auf dunkelblauem Grund mit hellblauer Umrandung. Direkt unter der READY-Meldung befindet sich ein kleines blinkendes Quadrat. Es ist der sogenannte Cursor, der anzeigt, daß der Computer auf eine Eingabe wartet, was er zu tun hat. Drücken Sie mehrmals die RETURN-Taste, werden Sie sehen, daß der Cursor auf dem Bildschirm nach unten wandert. Die erste Nachricht wird oben aus dem Bildschirm rollen. Der Cursor befindet sich nun auf Ihrem Schirm links unten. Um ihn nach links oben zu schicken, drücken Sie die Taste CLR/HOME in der rechten oberen Ecke Ihrer Tastatur. Der Cursor springt sofort in die linke obere Ecke des Monitors. Wenn Sie dies probiert haben, sehen Sie , daß Ihr Computer und die Tastatur eine Einheit bilden. Wir werden gleich zur Tastatur zurückkehren. Lassen Sie uns aber zuerst Ihren Drucker, Ihr Diskettenlaufwerk und/oder Ihren Kassettenrecorder testen.

Um zu sehen, ob Ihr Drucker korrekt arbeitet, geben Sie folgendes Programm GENAU SO EIN:

Schreiben Sie zuerst das Wort NEW und drücken Sie RETURN. (mit <RETURN> ist die so markierte Taste gemeint.)

```
10 OPEN7,4<RETURN>
20 PRINT#7,"MEIN DRUCKER ARBEITET!"<RETURN>
30 CLOSE7<RETURN>
```

Vergewissern Sie sich, daß Sie das Programm genauso geschrieben haben, wie es hier oben steht. Wenn auch nur geringfügige Unterschiede bestehen, verbessern Sie diese so, daß die Texte präzise übereinstimmen. Geben Sie das Farbband und Papier in den Drucker. Schalten Sie den Drucker jetzt ein, geben Sie das Wort RUN und dann <RETURN> auf Ihrem Computer ein. Wenn Ihr Drucker richtig angeschlossen ist, wird er die Meldung "MEIN DRUCKER ARBEITET!" ausgeben. Wenn "SYNTAX ERROR" oder eine andere Meldung auf Ihrem Monitor erscheint, heißt das, daß das kleine Testprogramm falsch eingegeben wurde und Sie es nochmal neu versuchen müssen. Wenn Ihr System "aussteigt", - der Bildschirm wird dunkel, und es passiert nichts mehr - sehen Sie nach, ob der Drucker eingeschaltet ist und der Schalter auf der Rückseite auf "4" steht. Wenn er jetzt immer noch nicht funktioniert, schalten Sie den Drucker und den Computer aus und wiederholen die Schritte AUFSTELLEN DES COMMODORE-64 UND DER PERIPHEREN GERÄTE.

Laden des Betriebssystems von der Diskette

Vorausgesetzt, Ihr System arbeitet richtig, können Sie jetzt eine Diskette in Ihrem Diskettenlaufwerk VIC-1541 "laden". (Wenn Sie ein anderes Diskettenlaufwerk haben, lesen Sie diese Schritte im mitgelieferten Manual nach.) Ihr Betriebssystem (DOS) wird folgendermaßen in den Computer gebracht: Schalten Sie Ihr Diskettenlaufwerk ein, indem Sie den Schalter hinten am Laufwerk auf ON stellen. Zuerst leuchtet das rote Licht auf und Sie hören einige Geräusche, bevor es erlischt und das grüne Licht erscheint. Legen Sie nun eine LEERE SOFT-SEKTORIERTE Diskette (nicht die mitgelieferte TEST-DEMO) mit dem kleinen Einschnitt links und der beschrifteten Seite nach oben in das Laufwerk. Drücken Sie die Klappe nach unten und leicht nach vorn, bis Sie ein Klicken hören. Jetzt können Sie die Diskette formatieren.

(HINWEIS: Wenn Sie eine Diskette schon formatiert haben, sollten Sie dies NICHT nochmals machen, außer Sie möchten alle Programme von der Diskette entfernen.)

Geben Sie folgendes ein:

```
OPEN15,8,15<RETURN>
PRINT#15,"N0:MEINE DISKETTE,20"<RETURN>
```

Die Diskette wird sich jetzt für eine Weile drehen und am Ende stoppen. Sie müssen nun Ihre Diskette "initialisieren". Geben Sie dafür folgendes ein:

```
PRINT#15,"I0"<RETURN>
```

Das ist alles. Jedesmal, wenn Sie eine neue Diskette ins Laufwerk geben, müssen Sie diese wie oben angegeben initialisieren. Wenn Sie eine Diskette mit "N0:" formatiert haben (N für NEW, aber nicht jenes NEW, das Sie benutzen, um den Speicher zu löschen!), brauchen Sie es nicht nocheinmal zu machen. Sie melden jedoch jede Diskette an, nachdem Sie diese ins Laufwerk gelegt haben.

WARNUNG!

Der Ausdruck "initialisieren" bedeutet für Ihr Diskettenlaufwerk: "fertig machen für den Gebrauch". Bei anderen Computern versteht man unter "initialisieren" das Formatieren einer Diskette. Erinnern Sie sich: eine Diskette wird einmal formatiert, aber nach jedem Einlegen ins Laufwerk initialisiert. Das "Initialisieren" der Diskette in Ihrem Commodore-Laufwerk löscht keines der vorhandenen Programme. "Formatieren" mit dem "N"-Kommando löscht die Programme auf der Diskette. Wenn Sie auf die kleine quadratische Einkerbung der Diskette eine "Schreibschutzmarke" kleben, ist diese vor zufälligem Löschen geschützt. Eine "schreibgeschützte" Diskette kann weder formatiert, noch überschrieben werden. Wenn Sie also eine Diskette haben, die Sie keinesfalls überschreiben möchten, vergewissern Sie sich, ob sich auf der Einkerbung ein Schreibschutzaufkleber befindet. Formatieren Sie niemals Ihre TEST/DEMO-Diskette, die mit Ihrem Diskettenlaufwerk geliefert wird.

Um nachzusehen, ob alles korrekt arbeitet, geben Sie folgendes ein:

```
LOAD"$",8<RETURN>
```

und wenn das rote Licht aus ist

```
LIST<RETURN>
```

Sie werden jetzt den Namen Ihrer Diskette auf dem Monitor sehen. Falls Sie bereits Programme abgespeichert haben, werden diese zum Inhaltsverzeichnis (Directory) hinzugefügt und namentlich aufgelistet, sobald Sie das "$" Kommando und LIST, wie eben gezeigt, eingeben. (Für detailliertere Angaben zum Gebrauch Ihres Diskettenlaufwerks sehen Sie im Kapitel 9 nach.)

Um ein Programm von der Diskette zu holen, laden Sie es und lassen es dann ablaufen. Nehmen Sie Ihre TEST/DEMO-Diskette und laden Sie eines der Programme, indem Sie folgendes eingeben:

```
LOAD"<PROGRAMMNAME>",8
```

Auf dem Bildschirm wird zuerst angezeigt, daß das Programm gesucht wird. Dann wird es geladen. Wenn das System READY anzeigt, geben Sie

```
RUN<RETURN>
```

ein, um das Programm auszuführen.

Laden und Ausführen von Programmen vom Recorder

Die Prozedur des Ladens und Ausführens von Programmen vom Recorder ist sehr einfach. Die folgenden Schritte zeigen die Vorgehensweise:

1. Vergewissern Sie sich, daß der Recorder angeschlossen ist und das Band auf den Anfang zurückgespult ist. Wenn Sie ein Band mit Programmen besitzen, benutzen Sie dieses zum Test. (Eine Spielkassette <kein Steckmodul> ist dafür geeignet.) Wenn Sie kein Band mit einem Programm zur Hand haben, geben Sie folgendes Programm ein:

   ```
   NEW<RETURN>
   10 PRINT"<MEIN NAME>"<RETURN>
   20 END<RETURN>
   SAVE"ICH"<RETURN>
   ```

   Spulen Sie Ihr Band zurück und drücken Sie gleichzeitig die REC- und PLAY-Tasten auf Ihrem Recorder. Wenn das Band stoppt und READY auf dem Bildschirm erscheint, drücken Sie die STOP-Taste und spulen Sie das Band wieder zurück.

2. Schalten Sie Ihren Computer ein und geben Sie, nachdem der Cursor erscheint, folgendes ein:

   ```
   LOAD"<PROGRAMMNAME>"<RETURN>
   ```

3. Drücken Sie die PLAY-Taste Ihres Kassettenrecorders. Der Bildschirm wird für kurze Zeit dunkel und anschließend sehen Sie folgende Meldung:

SEARCHING FOR<NAME DES PROGRAMMS>
FOUND<NAME DES PROGRAMMS>
<PRESS THE COMMODOREKEY-LOWER LEFT HAND
CORNER OF YOUR KEYBOARD>

Der Bildschirm wird jetzt wieder dunkel und bringt anschließend die Meldung:

LOADING
READY

4. Von diesem Moment an ist Ihr Programm geladen und ablauffähig. Geben Sie das Wort RUN ein und Ihr Programm wird ausgeführt. Wenn Sie unser Beispielprogramm benutzen, erscheint jetzt Ihr Name auf dem Bildschirm. Spulen Sie Ihr Band zurück, so daß es für die nächste Aufnahme bereit ist.

(Hinweis: Auf dem COMMODORE-64 ist es notwendig, während des Ladeprozesses die COMMODORE-Taste zu drücken. Ihr Recorder Manual berücksichtigt eventuell diese Tatsache nicht, da Ihr Computer möglicherweiser neuer ist als das Manual des Kassettenrecorders.)

Überspielen von Band auf Diskette

Wenn Sie sowohl ein Diskettenlaufwerk als auch einen Kassettenrecorder besitzen und nicht die lange Ladezeit beim Band abwarten möchten (speziell, wenn Sie mehrere Programme nacheinander von Band ablaufen lassen möchten), warum übertragen Sie nicht Ihre Dateien vom Band auf die Diskette? Laden Sie nur das DOS, legen Sie eine formatierte Diskette ins Laufwerk, initialisieren Sie diese und laden Sie das Programm vom Band. Wenn Ihr Bandprogramm geladen ist, schreiben Sie einfach SAVE <"NAME DER DATEI>",8 und Ihr Programm befindet sich jetzt auf der Diskette und läßt sich schneller abarbeiten.

Programme im ROM

Wenn Sie vorgefertigte Programme kaufen, die in einem ROM gespeichert sind, stecken Sie diese nur in den Cartridge-Ausgang und schalten Sie Ihren Computer ein. Er wird das Programm automatisch starten.

1.3 Die COMMODORE-64 Tastatur

Tasten wie bei der Schreibmaschine

Wenn Sie die Schreibmaschinentastatur kennen, werden Sie die meisten dieser Tasten auf der COMMODORE-64-Tastatur wiederfinden. Zum größten Teil übernehmen diese Tasten die gleichen Funktionen wie die Ihrer Schreibmaschine. Wenn Sie das Wort COMPUTER eingeben und dabei die gleichen Tasten wie bei der Schreibmaschine drücken, erscheint das Wort "COMPUTER" auf Ihrem Monitor genau wie auf einem Blatt Papier. Nur die Groß- und Kleinbuchstaben

werden nicht wie bei Ihrer Schreibmaschine umgeschaltet. Auf dem COMMODORE-64 wechseln Sie den Modus Groß-Klein-Schreibung durch gleichzeitiges Drücken der "Commodore- Taste" (ganz links unten mit dem Commodore-Zeichen) und der SHIFT-Taste. Wenn Sie dies machen, können Sie auf der Tastatur wie auf Ihrer Schreibmaschinentastatur arbeiten. Wünschen Sie Großbuchstaben, drücken Sie einfach die SHIFT-Taste und den gewählten Buchstaben wie bei der Schreibmaschine. Der Bildschirm hat allerdings nur 40 statt der 80 Zeichen, die die meisten Schreibmaschinen aufweisen. Natürlich können Sie nicht einfach irgend etwas auf dem Bildschirm eingeben. Sie bekommen jedesmal einen "SYNTAX ERROR", wenn Sie die RETURN-Taste drücken, bis Sie ein richtiges Kommando eingeben. Ansonsten betrachten Sie die Tastatur Ihres Computers wie eine normale Schreibmaschinentastatur. (Hinweis: In den meisten unserer Beispielprogramme benutzen wir nur Großbuchstaben.)

Tasten, die Sie auf der Schreibmaschinentastatur nicht finden

Während ein Großteil der Tasten auf Ihrem COMMODORE-64 denen einer Schreibmaschinentastatur gleichen, gibt es doch auch einige andere, und es ist daher wichtig, ihre Funktion zu kennen.

COMMODORE TASTE (Commodore-Zeichen) - Diese Taste - sie befindet sich links unten auf der Tastatur - wird benutzt, um von reinen Großbuchstaben auf den Groß-Klein-Modus (zusammen mit der SHIFT-Taste) umzuschalten und um die, auf den Tasten links stehenden, Grafikzeichen auszugeben. Drücken Sie gleichzeitig die COMMODORE- und die "S"-Taste, und Sie erhalten das linke Grafikzeichen der Taste auf dem Monitor.

CTRL (CONTROL) - In der oberen linken Ecke Ihrer Tastatur ist die CTRL-Taste, genannt "Control-Taste". Durch Drücken der CTRL-Taste und einer der Farbtasten (Tasten mit den Nummern 1 bis 8) erhalten Sie die Möglichkeit, die Farbe Ihres Bildschirms zu wechseln. Versuchen Sie die CTRL-Taste zu halten und drücken Sie eine der Tasten mit den Nummern 1 bis 8. Sie sehen dann auf dem Monitor einen andersfarbenen Cursor. Wenn Sie die CTRL-Taste und die Taste mit der Nummer 9 drücken, erhalten Sie eine Inversdarstellung. Drücken Sie CTRL-9 (gleichzeitig die CTRL-Taste und die Nummer 9), und geben Sie einige Buchstaben ein. Probieren Sie

auch die CTRL-9 und dann die CTRL-3-Tasten und drücken Sie ein paarmal die Leertaste und Sie erreichen interessante Effekte. Um die Inversdarstellung wieder auszuschalten, drücken Sie einfach die RETURN-Taste oder CTRL-0. Sie verstehen jetzt vielleicht die Bedeutung der Markierung auf den Tasten 1 bis 0 (z.B. BLK auf der Taste "1"), die für die bestimmten Charakteristiken stehen und die Sie durch gleichzeitiges Drücken der CTRL-Taste benutzen.

RUN/STOP und RESTORE - Diese beiden Tasten liegen auf Ihrer Tastatur einander genau gegenüber, werden aber immer zusammen benutzt. Die RUN/STOP-Taste allein bewirkt eine Programmunterbrechung. Benutzen Sie die RUN/STOP-Taste und die RESTORE-Taste zusammen, bringen Sie das Programm wieder in den Normalzustand zurück. Wenn Sie z.B. unsere Mustervorschläge durchgehen und gerade eine andere Farbdarstellung haben als die hellblaue Grundfarbe, drücken Sie nur gleichzeitig die RUN/STOP- und die RESTORE-Taste, und Ihr Bildschirm wird gelöscht, um dann im Grundzustand wieder zu erscheinen. Wenn Sie jemals in Schwierigkeiten kommen und sich Ihr System "aufhängt", denken Sie daran, daß die Tasten RUN/STOP und RESTORE es wieder starten und die im Speicher befindlichen Programme nicht verloren gehen. Betrachten Sie diese Taste als "Paniktaste" oder "Reset"-Taste.

CLR/HOME (CLEAR HOME) - In der rechten oberen Ecke befindet sich eine sehr bedeutende Taste, die CLR/HOME-Taste. In der Computersprache bedeutet "HOME", den Cursor in die obere linke Ecke des Bildschirms zu bringen. CLR (Clear) bedeutet, daß der Bildschirm vorher gelöscht wird. Um diese Taste zu testen, drücken Sie mehrmals die RETURN-Taste, so daß sich der Cursor unten auf dem Monitor befindet. Drücken Sie jetzt die CLR/HOME-Taste und Sie werden sehen, wie der Cursor in die obere linke Ecke des Bildschirms springt. Und nun drücken Sie SHIFT-CLR/HOME: der Monitor leert sich und der Cursor steht immer noch oben in der gleichen Position. (Geben Sie ruhig etwas Text ein und probieren Sie den Unterschied der CLR/HOME- und der SHIFT-CLR/HOME-Taste.)

CRSR (CURSOR) - In der rechten unteren Ecke der Tastatur befinden sich die Cursortasten. Sie werden benutzt, um den Cursor auf dem Bildschirm in jede beliebige Position zu bringen, ohne das Bild zu bewegen. Die Pfeile auf den Tasten markieren die Richtung, einmal mit, einmal ohne Drücken der SHIFT-Taste. Probieren Sie zur Übung folgendes: Drücken Sie SHIFT-CLR/HOME und plazieren Sie anschließend den Cursor in die Mitte des Bildschirms, nur unter Zuhilfenahme der CRSR-Tasten. Haben Sie dies geschafft, können

Sie mit den Tasten sicher umgehen. Wenn Sie die Cursortaste in Anführungszeichen eingeschlossen innerhalb einer PRINT-Anweisung "schreiben", geschehen merkwürdige Dinge. Der Computer interpretiert diese Taste jetzt als ein auf dem Bildschirm zu druckendes Zeichen. Geben Sie als Beispiel folgendes ein:

```
PRINT"(PRESSEN SIE JETZT 10 X DIE CURSORTASTE)HALLO"
<RETURN>
```

Sie bekommen eine Reihe von inversen "Q's", wenn Sie die "CRSR nach unten" drücken. Nachdem Sie auch die RETURN-Taste betätigt haben, sehen Sie die Nachricht "HALLO" zehn Zeilen unterhalb der eingegeben Kommandozeile.

RETURN - Die RETURN-Taste ist wie die Taste für den Zeilenvorschub auf Ihrer Schreibmaschine. Sie wird oft auch als "Carriage Return" oder "CR" bezeichnet. Die Taste arbeitet analog zur Carriage-Return-Taste der Schreibmaschine, da der Cursor an die linke Seite des Bildschirms springt, nachdem Sie die Taste betätigt haben. Es gibt jedoch auch andere Bedeutungen der RETURN-Taste, die Sie noch während des Programmierens kennenlernen werden.

Arrow- und Pi-Taste - Ganz links oben auf der Tastatur finden Sie eine Taste, die einen horizontalen Pfeil ausgibt. Die vertikale Pfeiltaste auf der rechten Seite der Tastatur besitzt neben der Grafik- noch weitere Funktionen. In der Normalstellung können Sie mit dieser Taste Zahlen potenzieren. Geben Sie z.B. 2 ^ 2 und RETURN ein, so erhalten Sie "4" auf dem Bildschirm, d.h. den Wert 2 mit 2 potenziert. Jetzt geben Sie SHIFT-^ <RETURN> ein und Sie erhalten 3.14159265, den Wert für Pi.

INST/DEL - Diese Taste ist eine "Editier-Taste" für INSert (Einfügen) oder DELete (Löschen). In Kapitel 2 werden wir diese Funktion genauer erklären.

TASTATUR GRAFIK - An der Vorderseite der meisten Tasten befinden sich verschiedene Grafikzeichen. Diese Zeichen werden auf dem Bildschirm dargestellt, indem Sie das gewünschte Grafikzeichen zusammen mit der SHIFT- oder COMMODORE-Taste drücken. Die rechten Grafikzeichen auf jeder Taste werden mit der SHIFT-Taste, die linken Zeichen mit der COMMODORE-Taste betätigt. (Die rechten Grafikzeichen können nicht direkt ausgegeben werden, wenn Sie

sich im Groß-Kleinbuchstaben-Modus befinden.) Probieren Sie diese Grafikzeichen mit Hilfe der SHIFT- und der COMMODORE-Taste gleich aus.

FUNKTIONSTASTEN - Ganz rechts liegen die sogenannten "Funktionstasten". Sie sind von "f1" bis "f8" durchnumeriert. Diese Tasten werden durch spezielle Kommandos Ihres Programms betätigt. Da diese Tasten etwas komplizierter sind, werden sie in Kapitel 10 getrennt behandelt. Sie brauchen die Tasten im Moment noch nicht. Wenn Sie jedoch schon etwas Programmiererfahrung haben, können Sie in Kapitel 10 nachschlagen, wie diese Tasten im Programm verwendet werden.

EINIGE NEUE BEDEUTUNGEN FÜR BEKANNTE TASTEN - Einige der bekannten Tasten haben für den Computer eine andere Bedeutung, als vom Symbol her anzunehmen wäre. Viele dieser Tasten haben mathematische Funktionen. Im nächsten Kapitel werden wir illustrieren, wie diese Tasten eingesetzt werden, und sie im Detail besprechen. Deshalb sehen wir uns hier die Tasten nur ganz kurz an.

SYMBOL	BEDEUTUNG
+	Addieren
-	Subtrahieren
*	Multiplizieren (anders als konventionell)
/	Dividieren (anders als konventionell)
^	Potenzieren

Zusätzlich zu diesen neuen Darstellungen für mathematische Symbole werden noch andere Tasten in einer Weise benutzt, die Ihnen nicht gebräuchlich erscheinen wird. Wir werden später die genaue Bedeutung dieser Tasten erklären, führen jedoch unten eine Reihe davon auf, um Ihnen zu zeigen, welche Möglichkeiten Ihr COMMODORE-64 überhaupt aufweist.

SYMBOL	BEDEUTUNG
$	indiziert eine String-Variable
:	wird als "Endekennzeichen eines Statements" im Programm benutzt
%	indiziert eine variable Ganzzahl
?	Abkürzung des PRINT-Kommandos

Machen Sie sich keine Gedanken, wenn Sie die Bedeutung dieser Zeichen im Moment noch nicht verstehen. Sie sollen mit Hilfe der Tasten nur auf die "Computersprache" vorbereitet werden. Wenn Sie mit der Tastatur und der Bedeutung der einzelnen Tasten etwas vertrauter werden, können Sie diese ganz einfach einsetzen. Es reicht vorerst, wenn Sie sich der unterschiedlichen Bedeutung der Tasten bewußt sind.

1.4 Zusammenfassung

Das erste Kapitel war ein Überblick über Ihre neue Anlage. Sie sollten jetzt wissen, wie Sie die einzelnen Teile Ihrer COMMODORE-64 Anlage aufstellen und zum Arbeiten bringen. Sie sollten auch in der Lage sein, eine Diskette zu laden, zu initialisieren, sich das Inhaltsverzeichnis Ihrer Diskette anzulegen (CATALOG) und ein Programm von Diskette oder Band laufen zu lassen. Auch sollten Sie einige der grundlegenden DOS-Kommandos zum Manipulieren Ihrer Dateien kennen. Schließlich sollten Sie mit der Tastatur bekannt sein und wissen, was mit Cursor gemeint ist. Im Moment gibt es noch sehr viel zu lernen. Lassen Sie sich also nicht entmutigen, wenn Sie vieles noch nicht ganz verstehen. Während unserer weiteren Schritte werden Sie mehr und mehr dazulernen, und was Sie im Augenblick vielleicht noch verwirrt, wird bald klar sein. Haben Sie Vertrauen zu sich selbst, und Sie werden bald Dinge ausführen können, die Sie nie für möglich hielten.

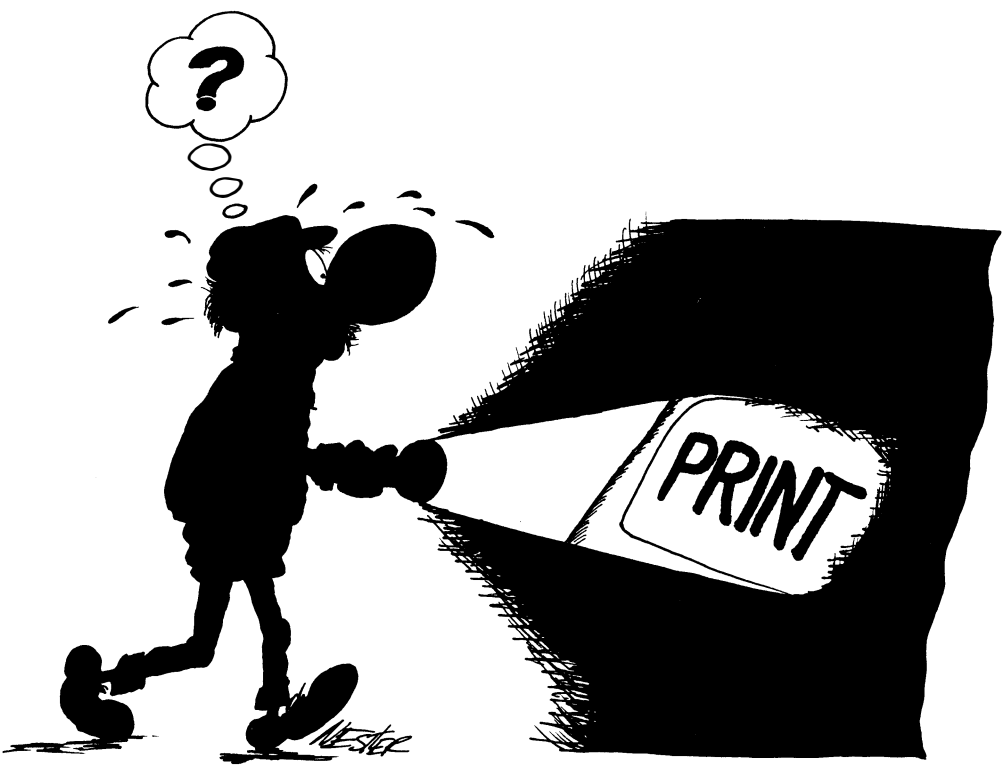
Im nächsten Kapitel werden Sie lernen, wie Sie Ihren COMMODORE-64 programmieren können. Es ist sehr wichtig, daß Sie die folgenden Beispiele eintippen und laufen lassen. Ebenfalls ist es empfehlenswert die vorgeschlagenen Änderungen nach dem ersten Versuch durchzuführen, damit Sie sehen, was Sie mit diesen geringfügigen Änderungen erreichen. Es gibt sowohl praktische, wie auch lustige (und verrückte) Programme in diesem Buch. Neben der Arbeit sollen Sie auch Freude an den Programmen haben.

2

Grundlagen

Einführung

Dieses Kapitel will Sie anweisen, Programme in der Sprache BASIC zu schreiben. COMMODORE-64-BASIC unterscheidet sich von einigen anderen Versionen dieser Sprache. Wenn Sie bereits etwas BASIC können, werden Sie diese Unterschiede bald herausfinden. Ist es für Sie jedoch eine neue Sprache, werden Sie feststellen, daß Programmieren in BASIC sehr einfach ist. Um zu starten, schalten Sie Ihren Computer ein, und wenn Sie auf dem Bildschirm das "READY"-Zeichen vorfinden, können Sie mit dem Programmieren beginnen. Wenn Sie auf dem Monitor etwas anderes sehen, drücken Sie gleichzeitig die RESTORE- und die STOP/RUN-Taste und geben Sie NEW ein, um den Speicher zu löschen.



2.1 PRINT

Das in BASIC wahrscheinlich am häufigsten benutzte Kommando ist PRINT. Wörter in Anführungszeichen, die dem PRINT-Kommando folgen, werden auf dem Monitor ausgegeben; ebenso Nummern und Variablen, wenn sie in einem PRINT-Kommando eingeschlossen sind. Es wird benutzt, um den Computer zur Druckausgabe auf dem Bildschirm oder Drucker anzuweisen; entweder innerhalb eines Programms oder im Kommandomodus. Sie werden sich jetzt bestimmt fragen, was der Unterschied zwischen Programmodus und Kommandomodus ist. Hierzu ein Beispiel:

Kommandomodus - Der Kommandomodus führt ein Kommando sofort aus, nachdem Sie RETURN drücken. Probieren Sie z.B. folgendes aus:

```
PRINT "DAS IST DER KOMMANDOMODUS"
DAS IST DER KOMMANDOMODUS

READY.
```

Sehen Sie, wie einfach das war? Geben Sie jetzt einige Zahlen aus, ohne diese in Anführungszeichen einzuschließen:

```
PRINT 6<RETURN>
PRINT 54321<RETURN>
```

Wie Sie sehen, können Sie Zahlen eingeben, ohne das Anführungszeichen zu benutzen. Wie Sie später bemerken werden, wird der genaue Wert der Zahlen wie ein "Bild" in den Speicher gestellt.

Programmodus - Dieser Modus stellt die Ausführung der Kommandos so lange zurück, bis Sie RUN eingeben. Alle Kommandos, die mit einer Zeilennummer auf der linken Seite beginnen, werden als Teile eines Programms behandelt. Probieren Sie folgendes:

```
10 PRINT "DAS IST DER PROGRAMMODUS"<RETURN>
```

Es geschieht nichts, richtig?

Geben Sie das RUN-Kommando. Ihr Bildschirm sieht jetzt so aus:

```
10 PRINT "DAS IST DER PROGRAMMODUS"
RUN
DAS IST DER PROGRAMMODUS

READY.
```

2.2 Ihr erstes Programm

Löschen des Bildschirms und Schreiben Ihres Namens

Lassen Sie uns ein Programm schreiben und zwei neue Kommandos lernen. Die neuen Kommandos sind CLR/HOME und END. Das CLR/HOME-Kommando löscht den Bildschirm und plaziert den Cursor oben links. Das CLR/HOME-Kommando wird durch gleichzeitiges Drücken der Taste CLR/HOME und der SHIFT-Taste eingegeben. Im Programmodus erscheint CLR/HOME als kleines Herz auf Ihrem Bildschirm. Keine Angst, es funktioniert. Das END-Kommando teilt dem Computer das Ende der auszuführenden Kommandos mit. Geben Sie im Kommandomodus das CLR/HOME-Kommando ein und verfolgen Sie was passiert. Lassen Sie uns jetzt ein Programm mit dem CLR/HOME-, END- und PRINT-Kommando schreiben. Drücken Sie von jetzt an die RETURN-Taste am Ende jeder Zeile. Für den Rest des Buches werde ich das <RETURN> nicht mehr aufführen, ausgenommen für Eingaben im Kommandomodus.

```
10 PRINT "<CLR/HOME>":REM EIN KLEINES HERZ ERSCHEINT
AUF IHREM BILDSCHIRM
20 PRINT "<MEIN NAME>"
30 END
RUN<RETURN>
```

Alles was Sie auf dem Bildschirm sehen sollten, ist Ihr Name, anschließend READY. Wir werden Ihnen jetzt zwei Kürzel vorstellen, die Ihnen helfen, Zeit bei der Eingabe und Platz im Speicher zu sparen. Erstens ist es möglich, mehrere Kommandos hintereinander in eine Zeile zu schreiben, wenn man sie durch Doppelpunkte ":" trennt. Ferner können Sie anstelle von PRINT das Fragezeichen "?" eingeben. Probieren Sie folgendes Programm, um dies zu testen:

```
10 ? "<CLR/HOME>"
20 ? "<MEIN NAME>":END
RUN<RETURN>
```

Es passiert genau dasselbe, aber Sie mußten nicht so viele Zeilen eingeben und das Wort PRINT nicht ausschreiben. Gut, oder? Noch etwas: Beginnen Sie Ihr Programm IMMER mit PRINT "CLR/HOME". Dies wird Ihnen helfen, Schwierigkeiten zu vermeiden, wenn Sie unterschiedliche Programme laufen lassen. Es wird auch Ausnahmen von dieser Regel geben. Aber größtenteils starten Sie Ihre Eingaben mit CLR/HOME, und dadurch beginnen Sie mit einem leeren Bildschirm anstatt mit einem großen Wirrwarr.

Am Anfang wird es sicherlich besser sein, den Doppelpunkt weniger häufig zu benutzen. Es ist auf diese Weise einfacher, die einzelnen Zeilen in einem Programm zu verstehen. Später, wenn Sie mehr Übung im Programmieren haben und Wege herausfinden möchten, um Speicherplatz zu sparen oder den Programmlauf zu beschleunigen, werden Sie den Doppelpunkt wesentlich häufiger benutzen. Schreiben Sie auch Ihre REM-Statements so kurz wie möglich. Dies sind Bemerkungszeilen und sie enden mit der nächsten Zeilennummer, die ein auszuführendes Kommando enthält. Sie dienen dazu, anderen den Programmablauf in den nachfolgenden Zeilen zu erklären, und sind auch für Sie selbst als kleine Erinnerung gut, um gleich zu sehen, was im Programm passiert. Um zu sehen, wie sie funktionieren, lassen Sie uns einige REM-Zeilen in ein kleines Programm einfügen.

```
10 PRINT "<CLR/HOME>":REM DIES LÖSCHT DEN SCHIRM
20 PRINT "<MEIN NAME>":END
30 REM DIESES PRÄCHTIGE PROGRAMM WURDE ERSTELLT
VON <MEIN NAME>
```

Wenn Sie jetzt RUN eingeben und das Programm laufen lassen, sehen Sie, daß die REM-Statements nicht auf dem Schirm erscheinen. Dennoch ist es sehr viel klarer, was ein Programm macht, wenn Sie hinter den Kommandos eine Erklärung in der Programmliste sehen.

2.3 Erstellen eines Programms

Benutzen von Zeilennummern

Nachdem wir nun schon ein kleines Programm geschrieben haben, sehen wir uns die Zeilennummern etwas genauer an. In Ihrem ersten Programm haben wir die Zeilennummern 10, 20 und 30 benutzt. Wir hätten auch die Nummern 0, 1 und 2 oder sogar 1000, 2000 und 3000 nehmen können. Es ist auch nicht wichtig, gleichmäßige Abstände zwischen den einzelnen Nummern zu verwenden. Ihr Programm arbeitet genau so gut mit den Zeilennummern 1, 32 und 1543.

Gewöhnlich werden wir unsere Programme in Zehnerschritten durchnumerieren und mit 10 beginnen. Sie werden zu Recht fragen, "Ist es nicht einfacher, Nummer 1, 2, 3, 4, 5, usw. zu nehmen?". Manchmal sicherlich, grundsätzlich aber nicht! Hier eine Begründung: Tippen Sie das Wort LIST <RETURN> ein und wenn Ihr Programm noch im Speicher ist, wird es jetzt auf dem Bildschirm erscheinen. Stellen Sie sich vor, Sie möchten eine Zeile zwischen die Zeilen 20 und 30 einfügen, um Ihre Adresse auszudrucken. Anstatt das Programm neu einzutippen, brauchen Sie jetzt nur eine Zeilennummer zwischen 20 und 30 (z.B. 25) zu wählen und diese einzugeben. Lassen Sie uns das einmal probieren; entfernen Sie aber zuerst das END-Kommando aus der Zeile 20.

```
25 PRINT "<MEINE ADRESSE>"
RUN<RETURN>
```

Aha! Sie sehen jetzt also Ihren Namen und Ihre Adresse auf dem Monitor. Sie mußten nur eine weitere Zeile eingeben, anstatt das ganze Programm zu wiederholen. Hätten Sie jedoch das Programm von 1 bis 10 durchnumeriert, wären Sie nicht in der Lage, zwischen den Zeilen 2 und 3 eine weitere Zeile einzufügen, wie es zwischen den Zeilen 20 und 30 möglich war. Sie müßten das gesamte Programm neu eintippen. Bei einem kleinen Programm ist das zwar kein Problem, wenn Sie aber 100- oder 1000zeilige Programme schreiben, werden Sie froh sein, zwischen den einzelnen Nummern etwas zusätzlichen Platz zu haben.

Auflisten Ihres Programms

Wie wir eben sahen, benutzen wir das Wort LIST, um unser Programm aufzulisten. Um dies besonders gut zu machen, geben Sie <SHIFT>-<CLR/HOME> und LIST <RETURN> ein und Sie bekommen das Listing auf einem leeren Bildschirm. Wenn Sie jedoch längere Programme schreiben, möchten Sie nie das gesamte Programm, sondern nur Teile davon auflisten. Lassen Sie uns die möglichen Optionen des LIST-Kommandos prüfen.

SIE SCHREIBEN	SIE ERHALTEN
LIST	Listing des gesamten Programms
LIST 20	nur die Zeile 20 wird aufgelistet (oder eine andere gewählte Nummer.)
LIST 20-30	alle Zeile von 20 bis inclusive 30 werden aufgelistet (oder eine andere Reihenfolge von Nummern.)
LIST -40	auflisten vom Beginn des Programms bis zur Zeile 40 (oder einer anderen gewählten Zeilennummer.)
LIST 40-	auflisten, ab der Zeile 40 (oder einer anderen gewählten Zeilennummer) bis an das Ende des Programms.

Versuchen Sie, verschiedene Teile Ihres Programms mit den möglichen Optionen aufzulisten, und sehen Sie nach, was Sie erhalten. Die folgenden Kommandos geben Ihnen ein Beispiel der unterschiedlichen Optionen:

```
LIST 25
LIST 20-
LIST -20
LIST 25-30
```

Erlauben wir uns einen Spaß?

Gewöhnlich benutzen Sie das LIST-Kommando im Kommandomodus nach der Erstellung des Programms. Sie können es aber auch im Programm verwenden. Fügen Sie nur zum Spaß folgende Zeile an Ihr Programm:

```
40 LIST
```

Geben Sie nun RUN ein und schauen Sie, was passiert. Ob Sie es glauben oder nicht: das ist eine sehr nützliche Anwendung, wie wir noch später in einigen Beispielen sehen werden. Im Moment ist es nur ein Spaß. Und nun zurück zum ernsthaften Stoff.

Abspeichern des Programms

Stellen Sie sich vor, Sie schreiben ein Programm; es läuft auch ganz richtig und nun schalten Sie Ihren Computer aus. Solange sich das Programm nur im RAM-Speicher befindet, wird es sich in NICHTS auflösen, und Sie sind gezwungen, es wieder einzutippen, um es laufen zu lassen. Glücklicherweise ist es eine einfache Angelegenheit, ein Programm auf Diskette abzuspeichern (SAVE). Nehmen wir unser Programm her, um das Abspeichern eines Programms auf Diskette zu probieren. Vergewissern Sie sich mit LIST, daß sich das Programm noch im Speicher befindet; andernfalls tippen Sie es erneut ein. Sehen Sie auch nach, ob sich im Diskettenlaufwerk eine formatierte und initialisierte Diskette befindet und geben Sie folgendes ein: (Wenn Sie sich mit dem Formatieren und Initialisieren der Diskette nicht ganz sicher sind, blättern Sie bitte zurück in Kapitel 1, wo beides erklärt wird.)

```
SAVE "0:MEIN PROGRAMM",8
```

Die Diskette wird sich jetzt drehen und das rote Licht an Ihrer Disketteneinheit leuchten. Wenn es erlischt, geben Sie LOAD "$",8 ein und wenn das READY erscheint, LIST. Sie werden überrascht sein, das Inhaltsverzeichnis zu sehen. Denn

MEIN PROGRAMM

steht jetzt im Inhaltsverzeichnis, d.h. das Programm wurde erfolgreich auf der Diskette abgespeichert.

Speichern des Programms auf Band

Um ein Programm auf Band zu speichern, legen Sie ein leeres Band in Ihren Recorder und spulen Sie es zurück. Drücken Sie gleichzeitig die RECORD- und PLAY-Taste Ihres Recorders und geben Sie SAVE "MEIN PROGRAMM" ein. Der Kassettenrecorder läuft an und die Nachricht OK erscheint zusammen mit SAVING MEIN PROGRAMM auf dem Bildschirm. Anschließend sehen Sie das READY. Ihr Programm ist jetzt auf Band gespeichert. Anders als bei der Diskette brauchen Sie keine "Ausgangsnummer" anzugeben (z.B. 8), da der COMMODORE-64 standardmäßig bei SAVE auf Band zugreift.

Zurückholen Ihres Programms

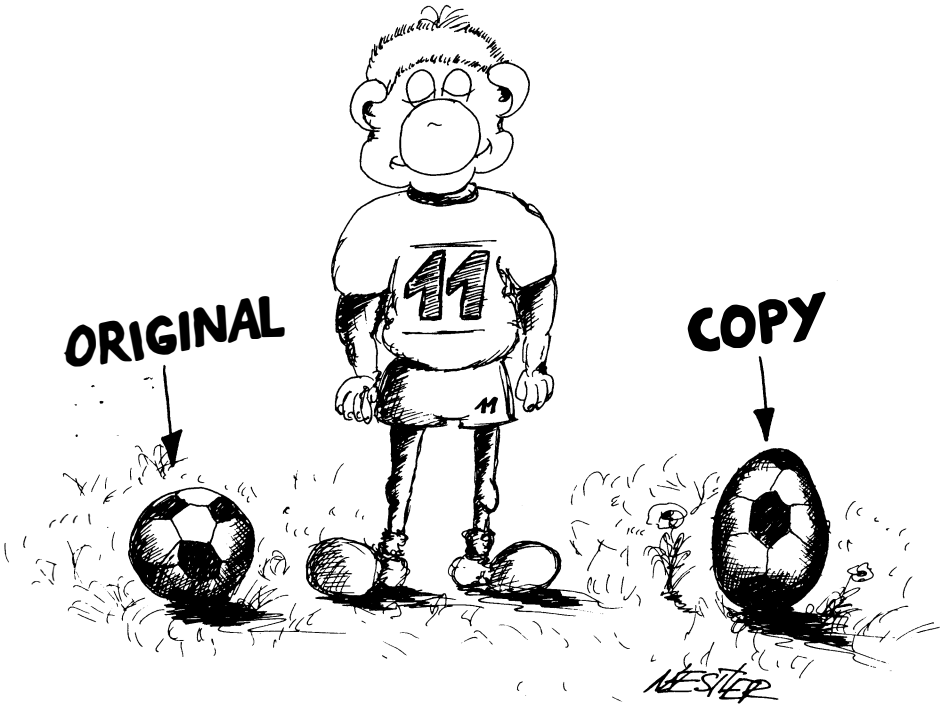
Um sich zu vergewissern, ob Sie ein Programm auf Band oder Diskette gesichert haben, schalten Sie Ihren COMMODORE-64 aus und gleich wieder ein. Tun Sie dies bitte. Initialisieren Sie Ihre Diskette, indem Sie OPEN 15,8,15 und PRINT#15,"INITIALIZE0" eingeben; anschließend LOAD"$",8 und LIST, um sich das Inhaltsverzeichnis anzusehen. Jetzt tippen Sie LOAD "0:MEIN PROGRAMM",8. Das Laufwerk dreht sich für eine Weile; Ihr Programm wird geladen und das READY erscheint erneut. LIST und RUN zeigt Ihnen, daß es dasselebe Programm ist, das Sie mit SAVE abgespeichert haben. Wenn es dasselbe Programm ist, haben Sie einen Beweis, daß Ihr Abspeichern erfolgreich war.

Besitzen Sie eine Bandeinheit, dann drücken Sie nur die PLAY-Taste des Recorders und geben Sie LOAD "MEIN PROGRAMM" ein. Das Band läuft an, um das Programm zu suchen, lädt es und anschließend meldet sich Ihr Computer mit dem READY. Mit LIST und RUN sehen Sie das gleiche Programm, das Sie gesichert haben.

Ein Sicherungstip

Wenn Sie beginnen, längere Programme zu schreiben - aber auch nur bei einigen Zeilen -, sollten Sie das Programm auf Band oder Diskette sichern. Übrigens, wenn Ihre Katze zufällig mal über die Tastatur läuft und dabei den Computer ausschaltet, ist somit Ihr Programm nicht verloren und Sie brauchen nicht auf die Katze loszugehen. Sie schützen so beides, Programm und Katze.

Da Sie nun Programme abgespeichert und wieder geladen haben, lassen Sie uns einen neuen netten Trick versuchen. Erinnern Sie sich, daß Sie Ihr Programm unter dem Namen MEIN PROGRAMM abgespeichert haben? Lassen Sie uns den Inhalt der Datei ändern. Geben Sie erst folgendes ein und listen Sie dann das Programm wieder auf:




```
27 PRINT "<MEINE ADRESSE>"
```

Ihr Programm unterscheidet sich jetzt von dem Programm, das Sie unter dem Namen MEIN PROGRAMM abgespeichert haben, bevor Sie Zeile 27 hinzufügten. Tippen Sie nun

```
SAVE "§0:MEIN PROGRAMM",8<RETURN>
```

Löschen Sie nun den Bildschirm, laden Sie es, und listen Sie es auf. Wie Sie sehen, ist die Zeile 27 jetzt ein Teil von MEIN PROGRAMM. Um ein Programm zu verändern, löschen oder fügen Sie Zeilen hinzu und speichern es unter dem gleichen Namen wieder ab, indem Sie "§" vor der "0" eingeben. Wie auch immer, seien Sie vorsichtig. Ganz gleich, welches Programm sich gerade im Speicher befindet, es wird unter dem eingegebenen Namen gesichert. Wenn Sie beispielsweise PROGRAMM A auf Ihrer Diskette haben, PROGRAMM B eintippen und unter unter dem Namen PROGRAMM A abspeichern, wird PROGRAMM A zerstört, und das abgespeicherte Programm ist das eigentliche PROGRAMM B. Auch wenn Sie ein sehr wichtiges Programm

haben, machen Sie davon eine "Sicherungskopie" (BACKUP). Wenn Sie z.B. Ihr erstes Programm unter dem Namen MEIN PROGRAMM und MEIN PROGRAMM BACK-UP abspeichern, haben Sie zwei Dateien mit ein und demselben Programm. Um dies auch wirklich sicher zu machen, nehmen Sie dazu zwei unterschiedliche Disketten.

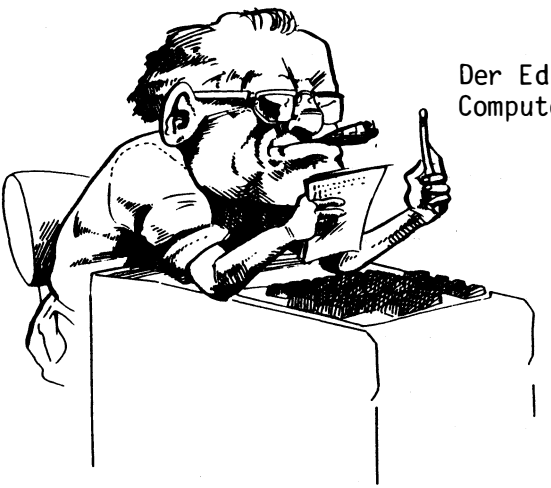
Ich habe Sie gewarnt

Früher oder später wird Ihnen folgendes passieren: Sie haben mehrere Disketten, formatierte und Programmdisketten. Sie nehmen die falsche Diskette (oder Band), eine Programmdiskette. Sie ist nicht schreibgeschützt, und nachdem Sie die Diskette oder das Band neu formatiert und somit alle Ihre wertvollen Programme gelöscht haben, werden Sie "!&$# ?!%&" schreien und sich die Haare raufen. Sie glauben das nicht, bis es einmal passiert ist. Um sich davor zu schützen, MACHEN SIE BACK-UP'S. Nehmen Sie Ihre Originale und bewahren Sie diese außer Reichweite auf. Falls Sie jetzt eine Diskette oder ein Band irrtümlich zerstört haben, machen Sie einfach noch eine COPY Ihrer Originaldiskette.

2.4 Verwendung des Editors

Fehlermeldungen und ihre Aufhebung

Es ist Ihnen vielleicht schon einmal passiert, daß Sie einen ?SYNTAX ERROR, ?SYNTAX ERROR IN 30 (Fehler in Zeile 30 oder einer anderen Zeile, in der ein Fehler entdeckt wurde) oder eine andere Fehlermeldung, wie z.B. REDO FROM START (die Ihnen mitteilt, daß etwas fehlt) angezeigt bekamen. Dies geschieht im Komandomodus, sobald Sie RETURN drücken, und im Programmodus, sobald Sie RUN eingeben. Abhängig vom Fehler bekommen Sie eine unterschiedliche Fehlermeldung. Abhängig von unterschiedlichen Operationen werden wir im Verlauf der nächsten Abschnitte verschiedene Fehlermeldungen kennenlernen. Im Moment werden wir uns die Möglichkeit näher ansehen, wie wir Fehler in Programmzeilen genau so schnell beseitigen, wie sie entstehen. Der Prozeß wird als "Editieren" des Programms verstanden.



Der Editor in Ihrem Computer!

Löschen von Zeilen

Die einfachste Art des Editierens besteht aus dem Einfügen und Löschen von Zeilen. Lassen Sie uns ein fehlerhaftes Programm schreiben und dieses dann verbessern.

```
NEW<RETURN>
10 PRINT "<CLR/HOME>"
20 PRINT "SOLANGE ETWAS FALSCH"
30 PRINT :"LAUFEN KANN":REM FEHLERZEILE
40 PRINT "ES GEHT"
50 END
RUN<RETURN>
```

Wenn das Programm genau wie oben eingegeben wurde, bekommen Sie ?SYNTAX ERROR IN 30. Tippen Sie nun:

```
30<RETURN>
LIST<RETURN>
```

Wo ist die Zeile 30? Sie haben eben gelernt, eine Zeile zu löschen. Immer wenn Sie nur eine Zeilennummer eingeben, löschen Sie diese Zeile. Wir haben auch schon gelernt, eine Zeile einzufügen. Geben Sie daher folgendes ein, um das Programm zu verbessern:

```
30 PRINT "LAUFEN KANN"
```

Wenn Sie nun das Programm laufen lassen, sollte alles in Ordnung sein. Der Fehler war der Doppelpunkt zwischen dem PRINT-Befehl und den Worten, die gedruckt werden sollten. Ein anderer Weg, das Programm zu verbessern, ist, die Zeilennummer neu einzugeben, ohne diese vorher zu löschen. Ich wollte jedoch gleichzeitig das Löschen von Programmzeilen vorführen.

Benutzen des Commodore-64 Editors

In Ihrem COMMODORE-64 steckt ein wirklicher Editor. Um zu sehen, wie Sie damit arbeiten, geben wir nochmals ein fehlerhaftes Programm ein und verbessern dieses. Geben Sie folgendes Programm ein, und lassen Sie es ablaufen:

```
NEW
10 PRINT "<CLR/HOME>"
20 PRINT "WENN ICH ETWAS FALSCH MACHE"
30 PRINT "KANN ICH": ES VERBESSERN: REM FALSCHE ZEILE
40 END
RUN<RETURN>
```

Richtig, Sie bekamen einen ?SYNTAX ERROR IN 30. Anstatt die Zeile neu einzugeben, probieren Sie folgendes;

1. Listen Sie das Programm.

2. Drücken Sie SHIFT und CRSR (vertikaler Cursor - die linke Cursortaste unter der RETURN Taste), und führen Sie den Cursor zur Zeile 30.

3. Benutzen Sie jetzt die rechte Cursortaste und bringen Sie den Cursor direkt auf den ersten Doppelpunkt.

4. Drücken Sie INST/DEL, bis das Anführungszeichen und der Doppelpunkt hinter ICH verschwinden.

5. Drücken Sie die rechte Cursortaste, bis der Cursor direkt auf dem Doppelpunkt steht. Jetzt drücken Sie SHIFT INST/DEL und sehen Sie, wie der Cursor eine Stelle nach rechts springt.

6. Tippen Sie nun ein Anführungszeichen hinter das "N" von "VERBESSERN" in die Leerstelle, die der Editor durch das INSert eingefügt hat. Drücken Sie RETURN und Sie haben die Zeile verbessert.

Listen Sie nun das Programm noch einmal auf: Zeile 30 sollte jetzt korrekt sein. Lassen Sie es laufen. In dem Statement sollten Sie jetzt "WENN ICH ETWAS FALSCH MACHE, KANN ICH ES VERBESSERN" sehen. Aber es gibt noch mehr zu lernen. Geben Sie folgendes Programm ein: (Erinnern Sie sich, daß Sie im COMMODORE-64-BASIC das ? anstelle des PRINT verwenden können? Wenn Sie das Programm vor dem RUN auflisten, werden Sie sehen, daß anstelle des ? plötzlich PRINT-Statements stehen.)

```
10 ?"<CLR/HOME>"
20 ?"MANCHMAL MAG ICH LANGE, LANGE, LANGE,
LANGE ZEILEN": WUH!
30 ?"UND MANCHMAL MAG ICH KURZE ZEILEN"
40 END
LIST<RETURN> (Beobachten Sie, was mit den ? passiert.)
RUN<RETURN>
```

Nun, nachdem Sie das Programm gestartet hatten, ging es schief. Das Problem steckte in dem Wort WUH!, weil es ohne PRINT-Anweisung oder Anführungszeichen hinter dem Doppelpunkt stand, auch keine REM-Zeile darstellte und die Zeile beendete. Um dies zu verbessern, bringen Sie den Cursor mit Hilfe der CRSR-Taste in die Zeile 20, d.h. in die fehlerhafte Zeile. Um die Sache zu vereinfachen, entfernen Sie das zweite Anführungszeichen, lassen den Doppelpunkt stehen und fügen hinter dem Wort WUH! ein Anführungszeichen ein. Da der Doppelpunkt nun innerhalb der Anführungszeichen steht, wird er als Teil des PRINT-Befehls ausgegeben und als Endekennzeichen des Befehls ignoriert. Drücken Sie RUN und RETURN.

Lassen Sie uns nun ein Merkmal des COMMODORE-64-Editors ansehen, das eventuell Schwierigkeiten mit sich bringt. Geben Sie folgendes ein, OHNE RETURN ZU DRÜCKEN!!!!:

```
20 PRINT "ICH MAG MEINEN COMPUTEEEEER
```

Hoppla! Da ist ein Fehler, aber Sie haben die Zeile noch nicht beendet. Keine Panik. Drücken Sie nur (SHIFT) H-CRSR und bringen den Cursor auf die mehrfach eingegebenen "E's" zurück. Tippen Sie es nun korrekt ein. (H-CRSR ist der "horizentale Cursor" und V-CRSR der "vertikale Cursor".) Sie sehen dies auch, wenn Sie die CRSR-Taste drücken, ohne den Cursor zu bewegen, dann bekommen Sie vertikale inverse Linien oder Klammern. Mit dem vertikalen Cursor bekommen Sie große blaue Punkte und inverse "Q's". Was ist passiert!??

Nicht so stürmisch, Watson. Wie wir in Kapitel 1 erwähnt haben, gibt Ihnen der COMMODORE-64 die Möglichkeit, in Anführungszeichen eingeschlossene, invers dargestellte Zeichen auszugeben, die Sie durch drücken der CRSR-Taste erzeugen. Um Änderungen vorzunehmen, drücken Sie erst RETURN und benutzen Sie dann die CRSR-Taste, um den Cursor an die zu verbessernde Zeile zu bringen. Wie Sie sehen, arbeiten die CRSR-Tasten jetzt richtig, auch innerhalb der Anführungszeichen. (ANMERKUNG: Lassen Sie mich noch sagen, daß es einfacher gewesen wäre, ein paarmal die INST/DEL-Taste zu drükken, um die mehrfachen "E's" zu beseitigen. Auf diese Weise hätten Sie aber nicht gelernt, warum die CRSR-Tasten, eingeschlossen in Anführungszeichen, anders arbeiten.)

Achten Sie auf 'RUNDY'

Nach dem Editieren mit dem COMMODORE-64 habe ich oft RUN über die READY-Meldung eingegeben und somit "RUNDY" erhalten. Natürlich gibt das statt RUN jetzt einen ?SYNTAX ERROR. Bei manchen Computern werden nach der Eingabe von RUN alle folgenden Buchstaben vergessen. Achten Sie, gerade wenn Sie auf anderen Computern schon geübt sind, auf dieses RUNDY!

Weiter im Editieren

Bevor wir weitermachen, wollen wir uns noch etwas mit dem Editor beschäftigen. Wir werden das Einfügen von Zeichen und Nummern üben, sowie auch das Editieren von Zeichengruppen. Probieren Sie folgendes kleine Programm:

```
10 PRINT "CLR/HOME"
20 PRINT "JETZT IST ZEIT FÜR ALLE GUTEN MÄNNER";
30 PRINT "ZU KOMMEN, UM IHR LAND ZU SCHÜTZEN"
40 END
```

So weit, so gut. Sie wollten aber auch die Frauen in die Zeile 20 eingeben. Sie können jetzt die gesamte Zeile neu eingeben; einfacher ist es jedoch, nur UND FRAUEN hinter MÄNNER einzufügen. Ebenso stört es Sie vielleicht, wenn alles in Großbuchstaben eingegeben ist. Lassen Sie uns die Zeile verbessern und dabei Groß-/Kleinschreibung verwenden:

1. Drücken Sie gleichzeitig SHIFT und "COMMODORE", um Kleinbuchstaben zu bekommen.

2. Bringen Sie den Cursor mit Hilfe der CRSR-Tasten zur Zeile 30 und positionieren Sie ihn rechts vom ersten Anführungszeichen.

3. Drücken Sie die SHIFT- und die INST/DEL-Taste und schaffen Sie damit genügend Platz, um " und Frauen" einfügen zu können.

4. Um den Satz zu korrigieren, plazieren Sie den Cursor über das "j" in "jetzt" der Zeile 20, und drücken Sie SHIFT und "J", damit Sie als erstes Zeichen des Satzes einen Großbuchstaben haben. (Das gleiche machen Sie mit den Wörtern "zeit", "männer" und "land".)

Nach diesen Reparaturen haben Sie nun Groß- und Kleinbuchstaben. Wenn Sie jetzt RUN eingeben, sollten Sie lesen können:

Jetzt ist Zeit für alle guten Männer und Frauen zu kommen, um ihr Land zu schützen.

Sie ersparen sich viel Zeit und Ärger, wenn Sie mit dem Editor arbeiten, anstatt alle Fehler einzeln auszubessern. Als praktische Übung sehen Sie nachfolgend ein paar zu verbessernde Zeilen. Die erste der beiden gleichen Zeilen ist die falsch eingegebene, die zweite die korrekte Zeile. Da Kleinigkeiten einen großen Unterschied ausmachen, sind dort mehrere Änderungen auszuführen. Sie werden sehen, daß Sie gerade auf diese kleinen Fehler am häufigsten hereinfallen werden. Üben Sie diese Beispiele, bis Sie sich im Editieren sicher fühlen - Zeit, die Sie jetzt investieren, sparen Sie später.

Editierbeispiele

```
50 PRINT  SEIN ODER NICHT SEIN."
50 PRINT "SEIN ODER NICHT SEIN."
10 PRINT <CLR/HOME>
10 PRINT "<CLR/HOME>"
80 PRINT "EIN GUTER MANN IST SCHWER ZU FINDEN"
80 PRINT "EINE GUTE KRAFT IST SCHWER ZU FINDEN"
40 PRINT "<CLR/HOME>" PRINT "WIR SIND FERTIG!
40 PRINT "<CLR/HOME>":PRINT "WIR SIND FERTIG!"
```

Wenn Sie diese Zeilen ausgebessert haben, können Sie jede Zeile editieren. Wenn Sie erst mal den Bogen raus haben, ist es ganz einfach.

2.5 Grundlegende mathematische Operationen

Bis jetzt haben wir nur Text ausgedruckt, was sich nicht sehr von der Arbeit mit einer guten Schreibmaschine unterschieden hat. Lassen Sie uns nun ein paar einfache mathematische Rechenoperationen durchführen, um Ihnen zu zeigen, wie Ihr Computer rechnet. Geben Sie folgendes ein:

```
<CLR/HOME>
PRINT 2+2
```

Ihr Bildschirm sollte jetzt so aussehen:

```
PRINT 2+2
4
```

Großartig: der Computer kann addieren - aber das kann auch ein 10-DM-Taschenrechner und ein 7 Jahre altes Kind. Wer sagte denn, Computer seien clever? Der Programmierer ist derjenige, der clever ist. Also, versuchen wir ein etwas umfangreicheres Beispiel.

```
<CLR/HOME>
PRINT 7.87*123.65
```

Auch das kann Ihr Taschenrechner noch genau so, jedoch ein siebenjähriges Kind hätte schon einige Arbeit damit.

Je weiter wir fortschreiten, desto mehr Aspekte mathematischer Probleme können wir einfügen. Im nächsten Kapitel sehen Sie, wie Werte in Variablen hinterlegt werden und noch andere Dinge, die Ihren Taschenrechner schocken würden. Im Augenblick jedoch sehen wir uns erst das Format von mathematischen Operation an. Das "+" und "-" funktionieren genauso wie in der "normalen" Mathematik. Das "x" wird für Multiplikationen durch den "*"(Stern), und der "---" wird für die Division durch den "/"(Schrägstrich) ersetzt.

Wenn wir anfangen, komplexere mathematische Berechnungen durchzuführen, wird es nötig sein, uns eine wichtige Regel zur Lösung der Probleme anzusehen. Diese wird "Vorgehensweise" genannt. Abhängig von den Operationen, die wir durchführen, und dem Ergebnis, das wir erhalten möchten, benutzen wir die eine oder die andere Regel. Stellen Sie sich z.B. vor, Sie möchten die Summe zweier Zahlen mit einer dritten multiplizieren - die Summe von 15 und 20, multipliziert mit 3. Wenn Sie dies so eingeben,

```
3 * 15 + 20
```

erhalten Sie 3 multipliziert mit 15 und addieren dazu 20. Aber das wollten Sie nicht. Es ist durch die gegebene Vorgehensweise passiert - Multiplikationen haben Vorrang vor Additionen. Um sich an diese Regelung zu erinnern, können Sie das folgende kleine Programm eingeben, es ablaufen lassen und dann im Kommandomodus einige mathematische Operationen eingeben. Die Ergebnisse können Sie dann anhand des "Vorgehensplanes" am Bildschirm kontrollieren. (Dieses kleine Programm ist sehr nützlich. Sichern Sie es deshalb auf Diskette oder Band.)

```
10 PRINT "<CLR/HOME>"
20 PRINT "1.- (MINUSZEICHEN FÜR NEGATIVE ZAHLEN - KEINE
SUBTRAKTION)"
30 PRINT "2.^ (EXPONENT)"
40 PRINT "3.*/(MULTIPLIKATION UND DIVISION)"
50 PRINT "4.+ -(ADDITION UND SUBTRAKTION)"
60 PRINT "ALLE ANDEREN WERDEN GLEICHRANGIG BEHANDELT "
70 PRINT "VORGEHENSWEISE IST VON LINKS NACH RECHTS "
80 PRINT "IHR COMPUTER LÖST ZUERST DIE KLAMMERAUS-"
90 PRINT "DRÜCKE, BEI GESCHACHTELTEN KLAMMERN, VON "
100 PRINT "INNEN NACH AUSSEN, AUF UND MULTIPLIZIERT "
110 PRINT "DIE AUFGELÖSTEN KLAMMERAUSDRÜCKE DANN."
```

Probieren Sie ein paar kompliziertere Rechenoperationen, und sehen Sie nach, ob Sie das gewünschte Ergebnis erhalten.

Änderung der Prioritäten

Wenn Sie die Regel herausgefunden haben, nach der mathematische Operationen ablaufen, sehen Sie sich an, wie diese mathematischen Probleme sehr einfach organisiert werden. Durch EINKLAMMERN von zwei oder mehreren Zahlen haben Sie die Möglichkeit, die Priorität heraufzusetzen. Lassen Sie uns zu unserem Beispiel von vorne zurückkehren und die Multiplikation der Summe diesmal mit Klammern durchführen.

```
PRINT 3 * (15 + 20)
```

Als das Multiplikationszeichen noch Vorrang vor dem Additionszeichen hatte, erhielten wir als Ergebnis 3 mal 15, plus 20. Da jetzt die Operation innerhalb der Klammern zuerst durchgeführt wird, addiert Ihr Computer ZUERST 15 und 20, und multipliziert diese Summe dann mit 3. Wenn in einem Ausdruck mehr als ein Klammernpaar ist, löst er die innerste zuerst auf und arbeitet sich nach außen durch.

Das Klammern-Gefängnis

Um Ihnen zu helfen, sich die Regelung mit den eingeklammerten mathematischen Operationen zu merken, denken Sie dabei an ein mehrstöckiges Gefängnis. Jede Zelle repräsentiert einen Klammerausdruck, der von links nach rechts aufgelöst wird. Jede Operation ist ein "Gefangener", umgeben von Wänden aus Klammern. Um aus diesem Gefängnis auszubrechen, muß der Gefangene erst einmal aus seiner Zelle kommen. Dann geht der Gefangene nach rechts und befreit alle anderen Gefangenen aus ihren Zellen. Als nächstes brechen diese aus dem "Zellenblock" aus und schließlich in die Freiheit. Führen Sie einige Beispiele durch. Vielleicht fällt Ihnen auch noch ein besserer Vergleich ein.

Die folgenden Beispiele zeigen Ihnen einige Operationen mit Klammern.

```
PRINT 20 + 10 * (8 - 4)
PRINT (12.43 + 92) /3 ^(11 - 3)
PRINT 22 * 3.1415 * 22 * 3.1415
PRINT (9 + 3) / (15 - 5)
PRINT - 10/2 * ((12 + 7) + (8 - 4))
```

Versuchen Sie nun einige der Probleme in einem geeigneten Format mit Ihrem Computer zu lösen:

Multiplizieren Sie die Summe von 4,9 und 20 mit 15

Multiplizieren Sie 35 mit 35 und das Ergebnis mit Pi(SHIFT ^). Die vertikale Pfeiltaste/Pi befindet sich zwischen dem Stern und der RESTORE-Taste. (Sie wissen, daß die Formel den Kreisinhalt mit einem Radius von 35 berechnet. Um den Inhalt eines anderen Kreises zu berechnen, tauschen Sie nur den Wert 35 aus.) Pi (SHIFT ^) wird genau wie eine andere eingegebene Zahl behandelt, aber um Zeit zu sparen, benutzen Sie diese vorbelegte Taste. Ganz gut, was?

Addieren Sie die Beträge für Ihre Ferngespäche und dividieren Sie die Summe durch die Anzahl der getätigten Anrufe.

Sie erhalten damit die durchschnittlichen Kosten Ihrer Telefongespräche. Erinnern Sie sich, daß sie diese Berechnung in einer einzigen Anweisung durchführen können. Machen Sie das gleiche mit Ihrem Scheckheft, um die durchschnittlichen Ausgaben pro Scheck zu errechnen.

2.6 Zusammenfassung

Dieses Kapitel hat die meisten grundlegenden Aspekte des Programmierens dargestellt. Sie sollen nun in der Lage sein, Ihren COMMODORE-64-Editor zu handhaben und Kommandos im Kommandomodus und im Programmodus zu schreiben. Sie sollten auch mathematische Operationen manipulieren können. Wie auch immmer, Sie haben gerade erst begonnen, die Macht Ihres Computers zu entdecken und behandeln ihn deshalb im Augenblick noch eher wie einen hervorragenden Rechner anstatt wie einen Computer. Nichtsdestoweniger ist das, was wir in diesem Kapitel gelernt haben, als Grundstein für Ihr späteres Programmierverständnis zu sehen. Wenn es im Moment noch einiges gibt, was Sie nicht ganz verstehen, blättern Sie bitte die letzten Seiten dieses Kapitels nochmal durch, bevor Sie weiterarbeiten. Falls Sie nach einer Wiederholung immer noch kleine Schwierigkeiten haben sollten, geben Sie nicht auf. Sie werden diese später überwinden. Es ist aber wichtig, daß Sie beim Durcharbeiten der Abschnitte alle vorgegebenen Übungen ausprobieren.

Das nächste Kapitel zeigt uns den Bereich der Computerprogrammierung und vertieft wesentlich Ihr Verständnis für den COMMODORE-64. Wenn Sie schrittweise vorgehen, werden Sie über die "Kraft in Ihren Fingerspitzen" erstaunt sein und sehen, wie einfach Programmieren ist. Wir verlassen jetzt auch den Bereich der "Taschenrechner"-Kommandos und sehen uns die tatsächliche Arbeit des Computers an. Dies ist der Moment, in dem der Spaß beginnt.

3

Wir machen weiter

Einführung

Im letzten Kapitel zeigten wir Ihnen die Ausführung einiger Kommandos im Kommando- und Programmodus. Von nun an konzentrieren wir uns auf den Ausbau der Grundkenntnisse im Programmodus aus Kapitel 2 und versuchen verschiedene Kommandos in einem Programm zusammenzufassen. Den Kommandomodus benutzen wir dazu, Ihnen einfache Beispiele und die Arbeitsweise einzelner Kommandos zu zeigen. Noch ein Tip: da wir ab jetzt laufend mehr Kommandos lernen, wäre es gut, die kleinen Beispiele auf Diskette oder Band abzuspeichern, da dies für einen Rückblick und ein schnelles Auffinden von Beispielen nützlich ist. Wählen Sie dafür Dateinamen, die Sie an den Inhalt erinnern, wie z.B. Variablenbeispiel oder Unterroutinentechnik und denken Sie daran, daß jede Datei einen eigenen Namen braucht. Sie können Dateinamen auch durchnumerieren (z.B. Tabelle1, Tabelle2, usw.).

3.1 Variablen

In den variablen Kommandos liegt wahrscheinlich die wichtigste aller Computerfunktionen. Grundsätzlich ist eine Variable ein Symbol, das mehr als einen einzigen Wert beinhalten kann. Wenn wir z.B. sagen x = 10, weisen wir der Variablen "X" den Wert 10 zu. Probieren Sie folgendes:

```
X = 10<RETURN>
READY.
PRINT X<RETURN>
```

Ihr Computer antwortet:

```
10
```

Jetzt tippen Sie:

```
X = 55.7<RETURN>
READY.
PRINT X<RETURN>
```

Wir machen weiter

und Sie erhalten jetzt:

```
55.7
```

Bei jeder neuer Wertzuweisung sehen Sie nach der PRINT-Anweisung den zuletzt zugewiesenen Wert der Variablen. Geben Sie folgendes Beispiel ein:

```
X = 10<RETURN>
Y = 15<RETURN>
PRINT X + Y<RETURN>
```

Ihr COMMODORE-64 antwortet mit:

```
25.
```

Wie Sie sehen, können Sie Variablen auch für mathematische Probleme benutzen. Sie setzen die Variablen direkt, anstelle der Zahlen, ein. Lassen Sie uns im nachfolgenden kleinen Beispiel Variablen benutzen, um den Inhalt eines Kreises zu berechnen.

```
10 PRINT "<CLR/HOME>"
20  PI  = (SHIFT) ^ :REM BENUTZEN SIE DIE  "PI"  TASTE  LINKS
NEBEN DER RESTORE TASTE
30 R = 15:REM R IST DER RADIUS DES KREISES
40  PRINT PI * (R*R):REM DIES GIBT UNS PI MAL DIE FLÄCHE DES
RADIUS
50 END
```

Wenn Sie RUN eingeben, erhalten Sie die Fläche eines Kreises mit dem Radius 15. Ändern Sie nun den Wert der Variablen "R" in Zeile 30, können Sie problemlos die Fläche jedes gewünschten Kreises berechnen. Da in unserem Beispiel eine Zahl quadriert wird, können wir auch unser Zeichen für den Exponenten "^" benutzen. Ändern Sie Zeile 40 so:

```
40  PRINT PI * (R ^ 2):REM DIE GLEICHE TASTE WIE BEI PI ABER
OHNE SHIFT TASTE
```

Lassen Sie das Programm noch einmal ablaufen und kontrollieren Sie, ob Sie das gleiche Ergebnis wie vorher haben. Verändern Sie auch den Wert von R und berechnen Sie damit verschiedene Kreisinhalte.

Variablennamen

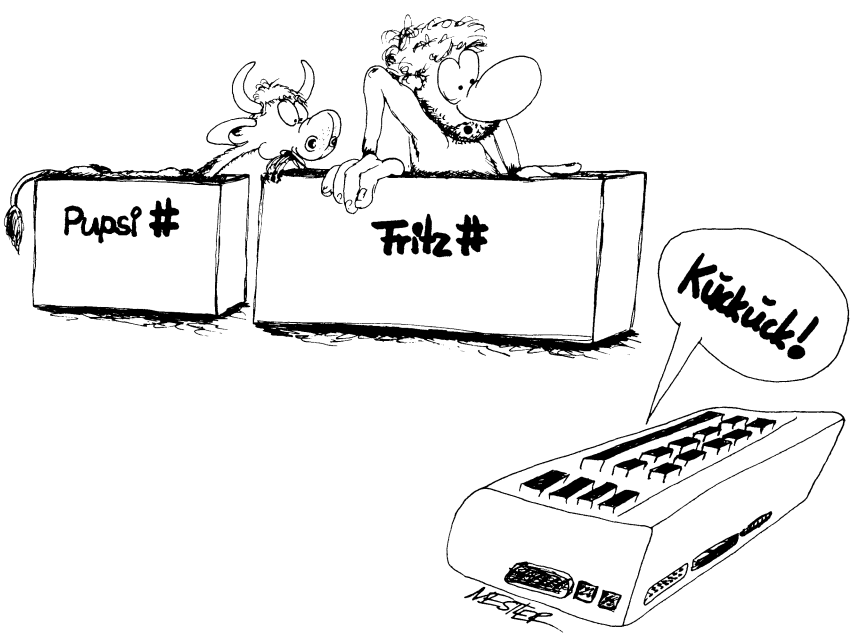
Wenn Sie einer Variablen einen Namen geben, beachtet der Computer nur die ersten beiden Zeichen. Nennen Sie z.B. eine Variable NUMMER, ist Ihr Computer nur an den Buchstaben NU interessiert. Testen Sie folgendes:

```
NUMMER = 63
PRINT NU
```

Sie erhalten 63, auch wenn Sie nur die ersten 2 Buchstaben der Variablen NUMMER eingegeben haben. Probieren Sie noch ein Beispiel:

```
NUMMER = 123
PRINT NUSS
```

Der Wert 123 wird ausgegeben, da immer noch nur die ersten zwei Buchstaben für den Computer interessant sind. So erhalten Sie den Wert 123, auch wenn die Namen der Variablen hinter den zwei ersten Buchstaben unterschiedlich sind.



Wir machen weiter

Am sinnvollsten scheint es, nur Variablen mit zwei Buchstaben zu verwenden. Wenn Sie in der Benutzung von Variablen geübter sind, ist das sicher eine gute Idee. Bei umfangreicheren Programmen ist es sehr nützlich, sprechende Variablennamen zu vergeben. Im folgenden Programm wird z.B. MITTEL als beschreibender Variablenname festgelegt:

```
10 PRINT "<CLR/HOME>"
20 A = 15: B = 23: C = 38
30 MITTEL = (A +B + C) / 3
40 PRINT MITTEL
50 END
```

Wäre dieses Programm hundert oder mehr Zeilen lang, wüßten Sie auch dann noch, wozu die Varable dient - sie errechnet einen "MITTEL"-Wert. Jetzt müssen Sie aber darauf achten, daß Sie nicht eine zweite Variable mit dem Namen MINDEST oder einem ähnlichen mit MI beginnenden Namen versehen.

Benutzen Sie zur Benennung der Variablen keine "reservierten Wörter" (z.B. Programmierkommandos) oder reservierte Variablen und achten Sie darauf, daß die erste Stelle ein Buchstabe ist. Es gibt nur die folgenden 3 reservierten Variablen: TI,TI$ und ST. Lassen Sie uns folgendes Beispiel durchgehen und sehen, was ein richtiger oder falscher Variablenname ist:

PRINT = 987 (Falsch, da PRINT ein reserviertes Wort ist.)

R1 = 321 (Gültiger Name, da erste Stelle ein Buchstabe.)

1R = 55 (Falsch, da das erste Zeichen kein Buchstabe ist.)

FORT = 222 (Falsch, da der Variablenname das reservierte Wort FOR beinhaltet.)

PR = 99 (Gültig, auch wenn das reservierte Wort PRINT ebenfalls mit PR beginnt; es wird nur ein Teil des reservierten Wortes im Variablennamen verwendet.)

TO = 983 (Falsch, da TO ein reserviertes Wort mit diesen zwei Buchstaben ist.)

TI = 99999999 (Falsch, da TI eine reservierte Variable für die Zeitausgabe darstellt.)

ADFEUDNDHFUFJCJKJDF = 10 (Gültiger Name, aber sinnlos.)

Es ist auch möglich, Werte an Variablen zu übergeben, indem man andere Variablen oder Variablenkombinationen und Zahlen nimmt. In unserem Beispiel mit der Variablen MITTEL definierten wir diese mit Hilfe anderer Variablen. Hier sind weitere Beispiele:

```
T = A * (B + C)
N = N + 1
SUM = X + Y + Z
```

Variablentypen

Variablen mit einfacher Genauigkeit

Bisher haben wir in unseren Beispielen nur "Fließkomma"-Variablen benutzt. Jede Variable, die mit einem Buchstaben beginnt und nicht mit einem Dollarzeichen ($) oder einem Prozentzeichen (%) endet, ist eine Variable mit einfacher Genauigkeit. Der Wert für eine Variable mit einfacher Genauigkeit kann zwischen + oder - 2.93873588E-39 und + oder -1.70141183E+38 liegen. Das "E" steht für die Exponentialdarstellung für sehr große Zahlen. Machen Sie sich darüber im Moment noch keine Gedanken. Fragen Sie jedoch einen Mathematiker, wenn Sie ein Ergebnis mit diesem Buchstaben in einem numerischen Resultat erhalten. Stellen Sie sich im Moment nur vor, daß Sie Zahlen in ihrem Standardformat von 0,01 bis 999.999.999 eingeben können. (Wenn Ihr Konto oder Ihre Einkommenssteuererklärung dieses wissenschaftliche Zeichen "E" enthalten, verlassen Sie besser das Land.) Denken Sie bei Variablen mit einfacher Genauigkeit nur daran, daß sie alle gebräuchlichen Zahlen, zusammen mit Dezimalstellen aufnehmen können.

Integer-Variablen

Integer-Variablen enthalten nur "Ganzzahlen" - d.h. Zahlen ohne Kommastellen. Sie sehen im folgenden einige Beispiele:

```
AB% = 345
K% = R% + N%
ADD% = ADD% + NUM%
WXY% = 88 + LR%
```

Die Werte für Integer-Variablen haben einen Bereich von -32767 bis +32767. Wie bei den Variablen mit einfacher Genauigkeit, werden nur die ersten zwei Buchstaben gelesen. Das "%"-Zeichen wird aber immer gelesen, unabhängig davon, wie viele Buchstaben davor stehen. So ist eine Variable mit dem Namen WA% dieselbe Variable wie WAX%. Jedoch eine Variable mit dem Namen ABC ist verschieden von der Variablen mit dem Namen ABC%; so können beide Variablen als einmalig angesehen und im gleichen Programm verwendet werden. Da sie aber einen kleineren Bereich als die Variablen mit einfacher Genauigkeit abdecken, haben Integer-Variablen ein begrenztes Anwendungsgebiet. Integer-Variablen brauchen jedoch weniger Speicherplatz und sind schneller in der Ausführung als Variablen mit einfacher Genauigkeit. Daher haben sie viele nützliche Anwendungsbereiche. Sie können genau wie Variablen mit einfacher Genauigkeit in mathematischen Operationen verwendet werden. Da sie aber keine Dezimalstellen mitabspeichern, sind sie bei Operationen, die Kommastellen erzeugen (z.B. die Division), mit Vorsicht einzusetzen. Probieren Sie einige der folgenden Operationen im Kommandomodus, um zu sehen, wie diese funktionieren:

```
A% = 15: B% = 21: C% = B% + A%: PRINT C%<RETURN>
36
LL% = 17: JJ% = LL% / 5: PRINT JJ%<RETURN>
3
Z% = -11: XY% = Z% + 51: PRINT XY%<RETURN>
40
```

String-Variablen

Stringvariablen sind bei der Formatierung der Ausgaben auf dem Bildschirm sehr nützlich. Genau wie Integer-Variablen und Variablen mit einfacher Genauigkeit werden sie durch das PRINT-Kommando auf dem Monitor angezeigt. Mit Stringvariablen können jedoch nicht nur Zahlen, sondern alle Zeichen, genannt "Strings" (Texte), ausgegeben werden. Stringvariablen werden durch ein Dollarzeichen ($) am Ende des Variablennamens gekennzeichnet. So sind z.B. A$, BAD$, GS$ UND PULL$ gültige Stringvariablen. (In der Computersprache verwenden wir den Ausdruck "String" anstelle des Dollarzeichens. So nennen wir unsere Beispiele "A String", "BAD String", usw.) Stringvariablen definiert man durch Einschließen des "Strings" in Anführungszeichen, genau wie bei unseren Texten, die wir mit PRINT ausgeben.

Lassen Sie uns einige Beispiele im Kommandomodus probieren:

```
ABC$ = "ABC": PRINT ABC$<RETURN>
GS$  = "POSSE": PRINT GS$<RETURN>
KAT$ = "KATZE": PRINT KAT$<RETURN>
NUMMER$ = "123456789": PRINT NUMMER$<RETURN>
B1$ = "5 + 10 + 20": PRINT B1$<RETURN>
```

Genau wie Integer-Variablen und Variablen mit einfacher Genauigkeit, benutzen die Stringvariablen nur die ersten beiden Stellen. Sie müssen daher mit einem Buchstaben beginnen und dürfen keine reservierten Wörter sein. Wie Sie in unserem Beispiel bemerkt haben, werden Zahlen in Stringvariablen nicht wie Zahlen, sondern als "Worte" oder "Nachrichten" behandelt. Bei der Ausgabe von B1$ erhielten Sie nicht "35" (die Summe von 5 + 10 + 20), sondern genau die Eingabe in Anführungszeichen: 5 + 10 + 20. Verwenden Sie Strings also nicht in der Mathematik. (In späteren Kapiteln werden wir einige Tricks zeigen, wie Sie Stringvariablen in Integer-Variablen oder numerische Variablen mit einfacher Genauigkeit umwandeln können. Im Moment betrachten Sie diese noch als Texte.)

Mit dem angesammelten Wissen schreiben wir jetzt ein Programm, welches Variablen beinhaltet. Das kleine Programm erlaubt Ihnen, die Ausgaben eines Schecks vom Guthaben zu subtrahieren und das neue Guthaben auszugeben. Dieses Programm wird später weiter ausgebaut, so daß Sie die Möglichkeit erhalten, damit Ihr Haushaltskonto zu führen.

Wir machen weiter

```
10 PRINT "<CLR/HOME>"
20 BALANCE = 571.88:REM JEDER ANDERE WERT IST MÖGLICH.
BALANCE(BA) IST EINE VARIABLE MIT EINFACHER GENAUIGKEIT
30 SCHECK = 29.95:REM IHRE LETZTE AUSGABE IM  COMPUTERLADEN.
SCHECK(SC) IST EINE VARIABLE MIT EINFACHER GENAUIGKEIT.
40 B$ = "IHRE ANFANGSBILANZ = DM "
50 C$ = "IHR SCHECK         = DM "
60 NB$ = "IHRE NEUE BILANZ   = DM ":REM B$,C$ UND NB$
SIND STRINGVARIABLEN
70 PRINT B$; BALANCE
80 PRINT C$: SCHECK
90 N = BALANCE - SCHECK
100 PRINT NB$: N
110 END
```

Da dies ein sehr langes Programm ist, vergewissern Sie sich, ob Sie alles richtig eingegeben haben. Ihr Computer ist sehr kritisch, wenn Sie Komma, Semikolon und Punkt verwechseln. Sichern Sie das Programm auch auf Diskette. Um damit zu spielen, verändern Sie die Werte in Zeile 20 und 30.

Lassen Sie uns kurz wiederholen, was wir gemacht haben:

1. Als erstes haben wir die Variablen mit einfacher Genauigkeit mit den Namen "BALANCE" und "SCHECK" definiert, welche Ihr Computer als BA und SC liest (nur die ersten zwei Buchstaben).

2. Dann haben wir die Stringvariablen B$, C$ und NB$, die wir als Label zur "Bildschirmformatierung" benutzen, definiert.

3. Schließlich haben wir alle unsere Informationen mit Hilfe unserer Variablen und der neuen Variable "N", die wir für die Differenz zwischen BALANCE und SCHECK definiert hatten, ausgegeben.

Beachten Sie bitte, wie wir den "OUTPUT" (Bildschirmausgabe) mit unseren PRINT-Anweisungen formatiert haben. Das Semikolon ";" zwischen den Variablen bewirkt folgendes: Erstens teilt es dem Computer das Ende der einen und den Anfang der nächsten Variablen mit, und zweitens erteilt es dem Computer die Anweisung, die zweite Variable direkt hinter der ersten auszugeben. Genauer gesagt, nimmt es die Stringvariable NB$

IHRE NEUE BILANZ = DM#

und gibt den Wert der Variablen mit einfacher Genauigkeit N hinter dem DM-Zeichen aus (genau auf die Stelle, die wir mit dem #-Zeichen gekennzeichnet haben). Später werden wir uns noch eingehender mit der Formatierung von Texten beschäftigen. Vorher sehen wir uns aber noch die Interpunktion bei der Textdarstellung an. Wir benutzen das Komma ",", das Semicolon ";" und den "Zeilenvorschub", um einfache Textdarstellung zu zeigen. Geben Sie das folgende kleine Programm ein:

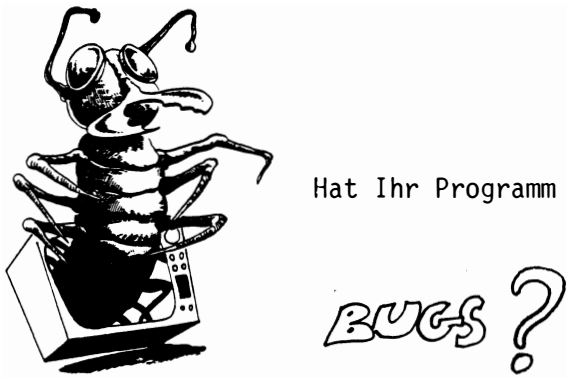
Wir machen weiter

```
NEW<RETURN>
10 PRINT "<CLR/HOME>"
20 A$ = "HIER": B$ = "DORT": C$ = "WO"
30 PRINT A$;: PRINT B$;: PRINT C$;:REM SEMICOLONS
35 PRINT
40 PRINT A$,: PRINT B$,: PRINT C$,:REM KOMMAS
45  PRINT:REM  EIN  ALLEINSTEHENDES  PRINT  ERGIBT  EINE
LEERZEILE
50 PRINT A$: PRINT B$: PRINT C$:REM NEUE ZEILEN
60 END
```

Lassen Sie das Programm nun ablaufen. Wie Sie sehen, erreichen Sie mit den kleinen Unterschieden in Zeile 30, 40 und 50 große Unterschiede in der Ausgabe am Bildschirm. Im ersten Beispiel wird alles ohne Zwischenraum hintereinander ausgegeben, beim zweiten wird der Text gleichmäßig über den Bildschirm verteilt, und beim dritten Beispiel wird jeder String in einer neuen Zeile dargestellt. Wie Sie im obenstehenden Programm sahen, stellen Semikolons Zahlen oder Strings direkt hintereinander dar. Benutzen wir hinter einer auszugebenden Variablen ein Komma, wird der Output in Vierergruppen über den Schirm verteilt. Verwenden wir dagegen Doppelpunkte oder neue Zeilennummern, erfolgt der Output in einer neuen Zeile. Eine PRINT-Anweisung für sich allein fügt ein vertikales "Line feed" (Zeilenvorschub) an. Probieren Sie folgendes Programm, um zu sehen, wie alleinstehende PRINTs verwendet werden können.

```
NEW<RETURN>
10 PRINT "<CLR/HOME>"
20 PRINT "IMMER, WENN SIE EINE ALLEINIGE PRINT ANWEISUNG";
:REM BEACHTEN SIE DIE STELLE DES SEMICOLONS
30 PRINT " EINGEBEN, BEWIRKT DIES EINEN ZEILENVORSCHUB."
40 PRINT
50 PRINT "SEHEN SIE WAS GEMEINT IST?"
60 END
```

Spielen Sie mit Kommas, Semikolon und Zeilenvorschub und verwenden Sie Variablen und Stringvariablen, bis Sie sich ganz sicher sind. Sie sind sehr wichtig und - falsch eingesetzt - oft die Ursache von Programmfehlern.



BUGS und Abstürze

Unter Programmierern tauchen oft die Ausdrücke "Debugging" und "Absturz" im Gespräch auf. "BUGS" sind Fehler in Programmen, die entweder einen ?SYNTAX ERROR erzeugen oder verhindern, daß Ihr Programm das macht, was Sie wollten. "Debugging" nennt man den Prozeß der Fehlerentfernung. Ein "Absturz" ist das, was Ihr Programm macht, wenn es auf einen Fehler stößt. Das ist Computersprache; wenn Sie die Ausdrücke in einer Unterhaltung gebrauchen, denken die Leute entweder, Sie wissen eine Menge über Computer oder Sie haben einen "Bug" in Ihrer Persönlichkeit.

Ist Ihr Programm abgestürzt?

3.2 Input und Output (I/O)

Input und Output, oft als I/O bezeichnet, ist die Eingabe von Informationen in den Computer und ihre Ausgabe. Gewöhnlich geben wir Informationen über die Tastatur ein, sichern sie auf Diskette oder Band und lesen die Informationen später von dort wieder ein. Die Ausgabe von Informationen erreichen wir über Bildschirm oder Drucker. Diese Vorgänge bezeichnet man als I/O. Bisher haben wir Informationen über die Tastatur eingegeben, entweder im Kommando- oder im Programmodus. Mit Hilfe der PRINT-Anweisung haben wir die Informationen auf den Monitor ausgegeben. Es gibt auch noch andere Wege, Informationen einzugeben, z.B. mit einer Kombination von Programm und Eingabekommandos. Lassen Sie uns einige dieser Wege betrachten, um unser SCHECKBUCH-Programm handlicher zu machen.

INPUT

Das INPUT-Kommando ist Bestandteil eines Programms. Es erwartet eine Eingabe von der Tastatur mit einem abschließenden <RETURN>. (Ein RETURN allein ist auch erlaubt, aber die Antwort wird dann als Leerstring interpretiert.) Das INPUT-Kommando kann daher im Kommandomodus nicht verwendet werden. (Nach Ausführung im Kommandomodus erhalten Sie die Fehlermeldung: ?ILLEGAL DIRECT ERROR.) Sehen wir uns ein einfaches Beispiel an:

```
NEW<RETURN>
10 PRINT "<CLR/HOME>"
20 INPUT X:REM X IST EINE NUMERISCHE VARIABLE - GEBEN SIE
DESHALB EINE NUMMER EIN
30 PRINT X
40 END
```

Wenn Sie RUN eingeben, sehen Sie einen leeren Bildschirm mit einem "?" und dem blinkenden Cursor dahinter, bis Sie eine Nummer eingeben. Anschließend gibt der Computer die gerade eingetippte Nummer aus. Wirklich interessant, wie?

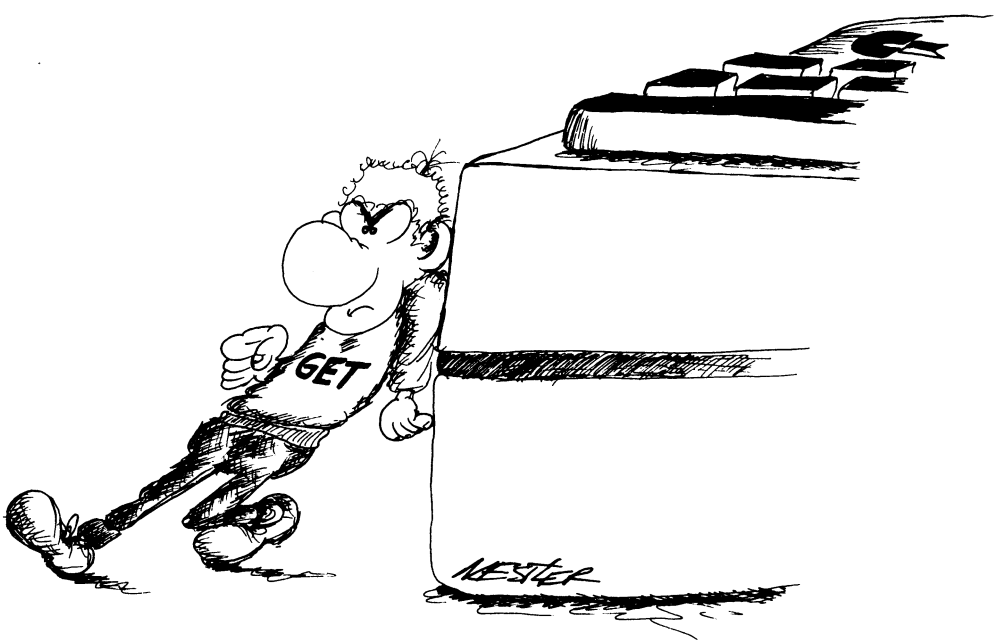
Lassen Sie uns die gleiche Information mit unterschiedlichem Format ausprobieren. Das Schöne an der INPUT-Anweisung ist, daß

Sie wie bei der PRINT-Anweisung Texte auf dem Bildschirm ausgeben können. Sehen Sie sich das folgende Programm an:

```
NEW<RETURN>
10 "<CLR/HOME>"
20 "GEBEN SIE IHR ALTER EIN";X
30 PRINT "<CLR/HOME>":PRINT:PRINT:PRINT
40 PRINT "IHR ALTER IST ";X
```

Geben Sie RUN ein. Sie sehen, daß die Darstellung ein bißchen interessanter ist. Beachten Sie auch, daß wir kein END-Kommando am Ende des Programms haben. Beim COMMODORE-64 ist es nicht erforderlich, eine END-Anweisung zu schreiben, aber grundsätzlich eine gute Idee. Wenn Sie anspruchsvolle Programme schreiben, sehen Sie, daß Sie innerhalb des Programms springen können und Ihre END-Anweisung dann mitten im Programm steht. Hier brauchen Sie die END-Anweisung, um ein klares Programmende festzulegen. Während wir das END-Kommando zwar im Augenblick nicht brauchen, ist es doch eine gute Angewohnheit, die man sich aneignen sollte.

Lassen Sie uns bei den INPUT-Anweisungen weitermachen.



Wir machen weiter

```
NEW<RETURN>
10 PRINT "<CLR/HOME>"
20 INPUT "GEBEN SIE IHREN NAMEN EIN --> ";NA$
30 PRINT
40 INPUT "GEBEN SIE IHR ALTER EIN   --> ";AG%
50 PRINT
60 INPUT "DRÜCKEN SIE <RETURN> UM FORTZUSETZEN";RT$
70  ?   "<CLR/HOME>":?:?:?:?:?:REM  BENUTZEN  SIE   DIESE
ABKÜRZUNG FÜR PRINT
80  PRINT  NA$;" IST ";AG%;" JAHRE ALT.":REM  PRÜFEN  SIE
GENAU DIE LAGE DER ANFÜHRUNGSZEICHEN UND DER DOPPELPUNKTE
IN DIESER ZEILE
90 END
```

Inzwischen sind wir soweit, daß Sie Informationen in Form von numerischen oder Stringvariablen eingeben können und wissen, wie Sie den Output gestalten. Wenn Ihre Programme länger und komplizierter werden, ist es wichtig, Ihre Stringvariablen und numerischen Variablen so zu verbinden, daß gut ersichtlich ist, was die Zahlen auf Ihrem Bildschirm bedeuten. Ein Computer, der den Bildschirm nur mit Zahlen füllt und bei dem Sie nicht wissen, was das bedeutet, wäre nicht sehr hilfreich. Zeile 60 hat das Format für eine Pause in Ihrem Programm. RT$ beinhaltet keine Informationen, bis Sie über die Tastatur etwas eingeben und RETURN drücken.

Informationen mit GET

Die GET-Anweisung ist ähnlich wie die INPUT-Anweisung. Sie wird ausgeführt, sobald Sie eine Taste drücken. Um zu sehen, wie Sie funktioniert, probieren Sie folgendes Programm. Beachten Sie, daß GET innerhalb einer Programmschleife liegt.

```
NEW<RETURN>
10 PRINT "<CLR/HOME>"
20 ?:?:?:?
30 PRINT "GEBEN SIE EINE NUMMER VON 1 - 9 EIN ";
35  GET  N:IF N < 1 OR N > 9 THEN 35:REM  BEACHTEN  SIE  DAS
FORMAT IN DIESER ZEILE
40 ?:?
50 PRINT "DRÜCKEN SIE EINE TASTE UM FORTZUSETZEN ";
55  GET K$:IF K$ = "" THEN 55:REM BENUTZT ALS PAUSE BIS EINE
```

```
TASTE GEDRÜCKT WIRD
60 ? "<CLR/HOME>":?:?:?:?
70 PRINT "IHRE NUMMER IST --> ";N
80 END
```

Beachten Sie, daß die GET-Anweisung ausgeführt wird, wenn Sie eine Taste drücken. Bei einer INPUT-Anweisung geben Sie zuerst Informationen ein und drücken dann die RETURN-Taste, bevor das Programm ausgeführt wird. Gut an der GET-Anweisung ist die schnellere Eingabe und Ausführung von der Tastatur aus. Das Problem liegt darin, daß Sie nur ein Zeichen eingeben können. Wenn Sie die falsche Taste drücken, haben Sie keine Chance, den Fehler zu berichtigen, wie es bei der INPUT-Anweisung möglich ist.

Das Lesen von DATA-Zeilen

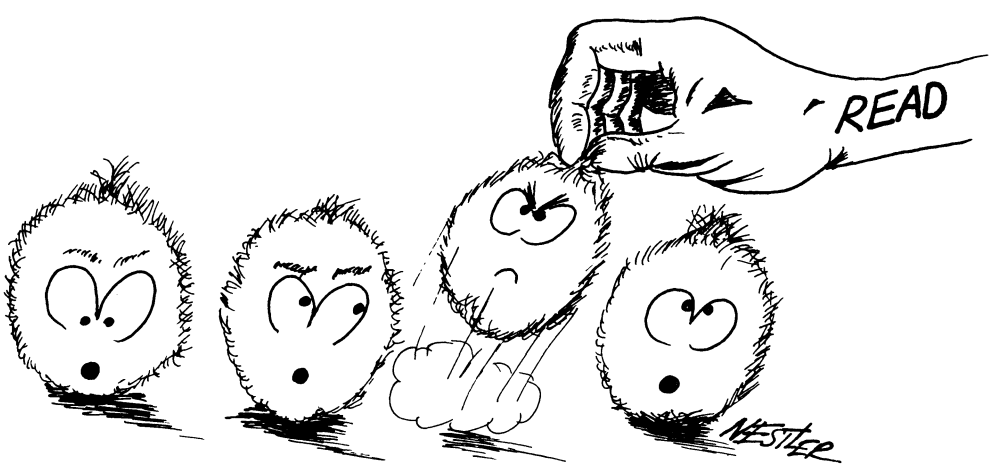
Der dritte Weg, Daten in ein Programm einzugeben, erfolgt mit Hilfe der DATA- und READ-Anweisung. Anstatt Daten über die Tastatur einzugeben, werden Daten von einem Teil des Programms in einen anderen eingelesen. Jede READ-Anweisung sucht nach Elementen in DATA-Anweisungen. Die READ-Anweisung ist mit einer Variablen verbunden, die in der nächsten DATA-Anweisung nachsieht und den numerischen Wert oder den String in der Variablen hinterlegt. Lassen Sie uns folgendes Beispiel ansehen:

```
NEW<RETURN>
10 PRINT "<CLR/HOME>"
20 READ NA$:REM LIEST DEN NAMEN
30 READ OC$:REM LIEST DIE BRANCHE
40 READ ST$:REM LIEST DIE STRASSE
50 READ SN:REM LIEST DIE HAUSNUMMER
60 READ PL:REM LIEST DIE POSTLEITZAHL
70 READ WO$:REM LIEST DEN WOHNORT
90 ?:?:?
100  REM BEGINN DER AUSGABE DER DURCH READ EINGELESEN WERTE.
ACHTEN SIE AUF GENAUE EINGABE.
110 PRINT NA$
120 PRINT OC$
130 PRINT ST$;" ";SN
140 PRINT PL;" ";WO$
150 END
```

```
1000 DATA PETER SCHMITT, HARD- UND SOFTWARE, KAISERSTR., 18
1010 DATA 8000, MÜNCHEN 21
```

In der DATA-Anweisung sind die verschiedenen Elemente, durch Komma getrennt, hinterlegt. Wenn Sie eines dieser Elemente entfernen oder ein Komma übergehen, kann einiges schieflaufen. Wenn eine READ-Anweisung z.B. eine numerische Variable (wie die PLZ) erwartet und einen String (wie z.B. Wohnort) einliest, erhalten Sie eine Fehlernachricht. Denken Sie bei der DATA-Anweisung an eine Reihe von Zahlen und Strings. Das erste Element der DATA Zeile wird entfernt, sobald das Programm an die READ-Anweisung kommt. Die nächste READ-Anweisung sucht nach dem obersten Element dieser Reihe, indem sie von links nach rechts vorgeht. Speichern Sie das Programm ab und lassen Sie uns anschließend einen Fehler einfügen.

Listen Sie das Programm auf, und vergewissern Sie sich, daß Sie das Programm im Speicher haben, bevor Sie folgendes eingeben:



```
85 READ EX$
```

Geben Sie jetzt RUN ein und Sie erhalten ?OUT OF DATA ERROR IN 85. Der Fehler tritt auf, da Sie eine READ-Anweisung ohne ausreichende DATA-Anweisungen (oder Elemente) haben. Achten Sie deshalb darauf, daß erstens genügend Elemente in den DATA-Anweisungen für Ihre READ-Anweisungen vorhanden sind, und beachten Sie zweitens, daß die Variablen in Ihren READ-Anweisungen mit den Elementen der DATA-Anweisung übereinstimmen (z.B. lesen numerische Variablen die Zahlen und Stringvariablen die Strings.) Um Ihr Programm zu berichtigen, geben Sie folgendes ein:

```
1020 DATA WORT
```

Jetzt haben Sie wieder etwas für Ihre READ-Anweisung. (Natürlich können Sie auch die Zeile 85 wieder löschen.)

Wenn ein Element einer DATA-Zeile in Anführungszeichen eingeschlossen ist, gehören alle eingeschlossenen Zeichen zu diesem String. Machen Sie z.B. folgende Änderung in Ihrem Programm, und geben Sie RUN ein:

```
145 EX$
1020 DATA "LEOPOLDSTR. 125,8000 MÜNCHEN 40, WEST-GERMANY"
```

Beides, Nummern und Kommas, werden von der READ-Anweisung als Stringvariablen akzeptiert, weil sie in Anführungszeichen stehen. Entfernen Sie diese jetzt und geben Sie nochmals RUN ein. Sie erhalten nun nur den Wert bis zum ersten Komma (LEOPOLDSTR. 125), haben aber keine Probleme mit der Zahl 125, da diese als String behandelt wird. (Der Wert 125 kann aus diesem Grund nun nicht in mathematischen Berechnungen verwertet werden.) Experimentieren Sie mit unterschiedlichen Elementen, um zu sehen was passiert. Plazieren Sie die DATA-Zeile auch einmal an einen anderen Teil des Programms. Sie finden bald heraus, daß Sie nicht an einer bestimmten Stelle im Programm stehen muß.

3.3 Schleifen mit FOR/NEXT

Die FOR/NEXT-Schleife ist eine der nützlichsten Operationen in einem BASIC-Programm. Sie gibt dem Anwender die Möglichkeit, den Computer anzuweisen, eine festgelegte Anzahl an Anweisungen, mit einer variablen Schrittweite zu durchlaufen und auszuführen, bis das festgelegte Endekennzeichen erreicht wird. Lassen Sie uns ein einfaches Beispiel dazu ansehen.

```
NEW<RETURN>
10 PRINT "<CLR/HOME>"
20 NA$ = "<IHR NAME>"
30 FOR I = 1 TO 10:REM BEGINN DER SCHLEIFE
40 PRINT NA$
50 NEXT I:REM ENDE DER SCHLEIFE
60 END
```

Geben Sie RUN ein. Sie sehen, daß Ihr Name 10 mal auf der linken Seite des Bildschirms angezeigt wird. Nun ja, nicht allzu eindrucksvoll, aber wir werden sehen, wie nützlich dies bald sein wird. Als erstes betrachten wir noch ein anderes Beispiel und sehen uns an, wie "I" nach dem Durchlaufen der Schleife aussieht.

```
NEW<RETURN>
10 PRINT "<CLR/HOME>"
20 FOR I = 1 TO 10
30 PRINT I
40 NEXT I
```

Wie Sie sehen, verändert sich der Wert von "I" während des Programmlaufs bei jedem Betreten der Schleife. Denken Sie bei einer Schleife an ein Kind in einem Karussell. Nach jeder Runde erhält das Kind einen goldenen Ring, zuerst einen, und am Ende hat es in unserem Beispiel zehn.

Da wir jetzt die Funktion der Schleife kennen, lassen Sie uns eine weitere Übung durchführen. Wir verändern dazu unser abgespeichertes SCHECKBUCHPROGRAMM. Wir benutzen dafür mehrere Variablen, vergeben aber zuerst für die FOR/NEXT-Schleife traditionell die Variable "I". Als zweites brauchen wir eine Variable, um die Anzahl der Schleifendurchläufe festzulegen. Wir benutzen dazu N%, eine Integer-Variable. Drittens brauchen wir Variablen für den Saldo, die Höhe des Schecks und den neuen Saldo. Da das Programm

etwas länger wird, sichern Sie es zwischendurch ab, etwa alle 5 Zeilen. Auf das Band sichern Sie immer nach 10 Zeilen.

```
NEW<RETURN>
10 PRINT "<CLR/HOME>"
20 CB$ = "SCHECKBUCH"
30 PRINT:PRINT:PRINT CB$
40 INPUT "WIE VIELE SCHECKS? --> ";N%
50 INPUT "WIE HOCH IST IHR GEGENWÄRTIGER SALDO? --> ";BA
60 REM BEGINN DER SCHLEIFE
70 FOR I = 1 TO N%
80 PRINT "IHR SALDO IST JETZT DM ";BA
90 PRINT "HÖHE DES SCHECKS #";I;"--> ";
100 INPUT CK:REM VARIABLE FÜR DEN SCHECK
110 BA = BA - CK:REM ENTHÄLT DEN LAUFENDEN SALDO
120 NEXT I:REM ZUM ANFANG DER SCHLEIFE
130 ?  "<CLR/HOME>":REM LÖSCHT DEN SCHIRM WENN ALLE  SCHECKS
EINGEGEBEN SIND
140 ?:?:?
150 PRINT "SIE HABEN JETZT DM ";BA;"AUF IHREM KONTO"
160 PRINT:PRIMT "DANKE UND KOMMEN SIE WIEDER"
170 END
```

Unser Scheckbuchprogramm wird jetzt schon einfacher im Gebrauch, worin der Vorteil des Computers liegt. Lassen Sie uns noch etwas anderes in Verbindung mit Schleifen betrachten.

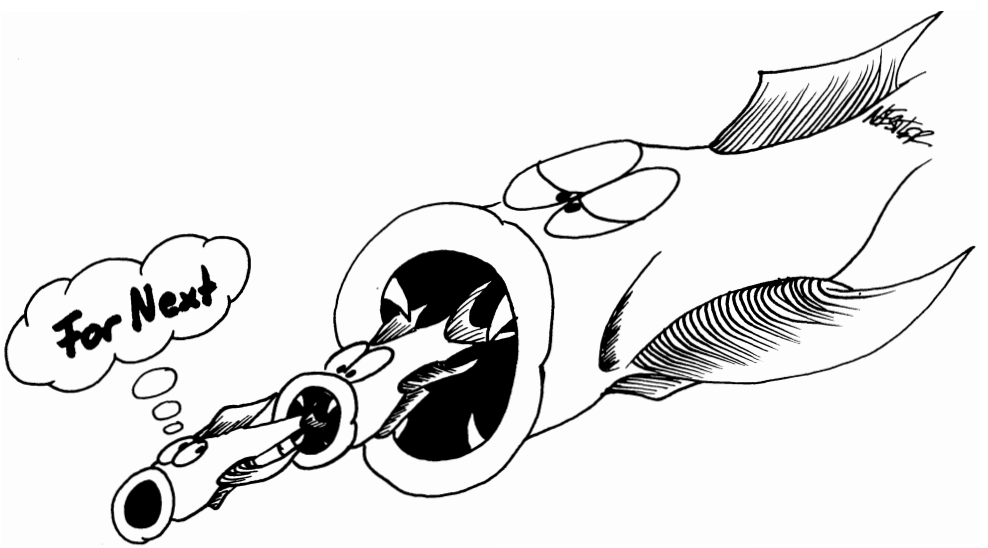
Eine Bagatelle

Wenn Sie sich einige Programme ansehen, merken Sie, daß die Variable "I" häufig in Programmschleifen verwendet wird. Sie können dafür auch jede andere Variable nehmen, gewöhnlich ist es jedoch "I". Ich war früher genau so neugierig wie Sie, warum Programmierer gerade den Buchstaben "I" aussuchen. Nach anstrengender Suche habe ich es auch herausgefunden. Das "I" war in FORTRAN (einer der ersten Programmiersprachen) die "Integer"-Variable und wurde für "DO-Schleifen" benutzt, da das schneller ging. Das "I" kann auch als "I" in "increments" (Zunahme) interpretiert werden. Ich sagte Ihnen ja, daß es eine Bagatelle ist...

Geschachtelte Schleifen

Bei manchen Anwendungen ist es erforderlich, eine weitere FOR/NEXT-Schleife in die bereits vorhandene einzufügen. Betrachten wir eine einfache Anwendung. Stellen Sie sich vor, Sie haben zwei Teams mit jeweils 10 Mitgliedern. Sie möchten nun ein Teamverzeichnis aufstellen und die Teamnummern (#1 oder #2) und Mitgliedsnummern (#1 bis #10) vergeben. Mit einer geschachtelten Schleife erreichen Sie dies im folgenden Programm:

```
NEW<RETURN>
10 PRINT "<CLR/HOME>"
20 FOR T = 1 TO 2:REM T FÜR TEAM#
30 FOR M = 1 TO 10:REM M FÜR MITGLIEDS#
40 PRINT "TEAM # ";T;"SPIELER # ";M
50 NEXT M
60 NEXT T
70 END
```



In einer geschachtelten Schleife ist es sehr wichtig, die Reihenfolge zu beachten. Die innerste Schleife (in unserem Beispiel die "M-Schleife") darf keine weitere FOR- oder NEXT-Anweisung beinhalten. Denken Sie bei der geschachtelten Schleife an eine Reihe einander fressender Fische. Das größte Fischmaul schluckt das nächstkleinere usw.

Betrachten Sie folgende Struktur der geschachtelten Schleifen:

```
FOR A = 1 TO N
  FOR B = 1 TO N
    FOR C = 1 TO N
      FOR D = 1 TO N
      NEXT D
    NEXT C
  NEXT B
NEXT A
```

Beachten Sie den Beginn (eine ausgeführte FOR-Anweisung) und das Ende (Treffen auf eine NEXT-Anweisung) der geschachtelten Schleifen. Haben Sie schon mal eine Reihe verschieden großer Kochtöpfe ineinander geschachtelt? Dies war möglich, da der äußere Rand des einen Topfes größer war als der des nächsten. Ebenso ist in geschachtelten Schleifen der "Umfang" einer Schleife "größer", als der "Umfang" der darin eingeschlossenen Schleife und "kleiner" als der "Umfang" der Schleife, in der diese liegt.

Vorwärts- und Rückwärtsschritte

Loops vergrößern sich normalerweise mit einer Schrittweite von "1". Es geht aber auch anders. Im folgenden Beispiel sehen Sie z.B. eine Schrittweite von "10":

```
NEW<RETURN>
10 PRINT "<CLR/HOME>"
20 FOR I = 1 TO 100 STEP 10
30 PRINT I
40 NEXT I
```

Dies erlaubt es Ihnen, den Zuwachs des Zählers selbst festzulegen. Sie können genauso eine Variable verwenden, sofern sie einen

numerischen Wert beinhaltet. Zum Beispiel:

```
NEW<RETURN>
10 PRINT "<CLR/HOME>"
20 K = 5:N = 25
30 FOR I = K TO N STEP K
40 PRINT I
50 NEXT
```

Geben Sie RUN ein. HALT!! werden Sie rufen. Sie haben in Zeile 50 einen "BUG" entdeckt, einen typischen Fehler. Nach dem Wort NEXT sollte ein "I" stehen, aber da ist keines; richtig? Nun, im COMMODORE-64-BASIC brauchen wir diese Variable nicht einzugeben. Sie sparen damit etwas Speicherplatz, wenn Sie die Variablen in den NEXT-Anweisungen nicht mit aufführen. Selbst in geschachtelten Schleifen ist die Angabe der Variablen hinter der NEXT-Anweisung nicht erforderlich, wenn Sie genügend NEXT-Anweisungen angeben. Es ist aber eine gute Programmierpraktik, die Variablen hinter den NEXT-Anweisungen anzugeben, da Sie den Verlauf der Schleifen besser verfolgen können.

Weiterhin ist es möglich, rückwärts zu gehen. Testen Sie das Programm:

```
NEW<RETURN>
10 FOR I = 4 TO 1 STEP -1
20 PRINT "ENDEPOSITION BEI # ";I
30 NEXT I
```

Wenn Sie kompliziertere (und nützlichere) Programme schreiben, sehen Sie, wie nützlich all diese unterschiedlichen Möglichkeiten des COMMODORE-64-BASIC sind. Oft werden Sie die praktische Verwendbarkeit der Kommandos nicht sofort sehen, wenn Sie diese aber später einmal einsetzen, werden Sie sich fragen, warum Sie nicht schon immer damit programmiert haben.

Falls Sie sich wundern sollten

Sie haben vielleicht gemerkt, daß die Zeilen in einem Loop eingerückt dargestellt werden. Haben Sie dies auf Ihrem COMMODORE-64 schon probiert, haben Sie herausgefunden, daß die Einrückung beim ersten Auflisten des Programms wieder verschwunden ist? Leider können Sie dies ohne spezielle Hilfsprogramme (Utilities) nicht vermeiden. Machen Sie sich deshalb keine Sorgen. Dies ist nur eine Festlegung beim Programmieren, die für Klarheit mit Hilfe eingerückter Zeilen innerhalb der Schleife sorgt, damit einfacher zu sehen ist, was das Programm macht. Die Darstellung hat keinen Einfluß auf den Programmablauf.

Zähler

Oft möchten Sie die Anzahl der Schleifendurchläufe nach ihrer Beendigung wissen und für eine spätere Verwendung zwischenspeichern. Wenn Sie z.B. ein Programm mit einer Schrittweite von 3 ablaufen lassen, wissen Sie vielleicht nicht genau, wie oft die Schleife durchlaufen wird. Um dies herauszufinden, benutzen Programmierer Zähler, die bei jedem Eintritt in die Schleife (gewöhnlich um +1) hochgezählt werden. Das folgende Programm zeigt Ihnen den Einsatz eines Zählers:

```
NEW<RETURN>
10 PRINT "<CLR/HOME>"
20 FOR I = 3 TO 99 STEP 3
30 PRINT I
40 N = N + 1:REM DIES IST DER ZÄHLER
50 NEXT I
60 ?:PRINT "DIE SCHLEIFE WURDE ";N;" MAL AUSGEFÜHRT."
```

Beim ersten Eintritt in die Schleife hatte der Zähler "N" den Wert 0. In der Zeile 40 wurde dann jeweils 1 zum gegenwärtigen Wert von "N" addiert (z.B. 0+1=1). Beim zweiten Durchgang der Schleife wurde 1 zum Wert 1 dazuaddiert. Am Ende der Schleife, in Zeile 50, betrug der Wert von N=2, und er behält diesen bis zum Austritt aus der Schleife. Nach Beenden aller Schleifendurchläufe teilt Ihnen N mit, wie oft die Schleife durchlaufen wurde. Natürlich zählen diese definierten Zähler nicht wirklich die Durchläufe und können deshalb um jeden beliebigen Wert hochgezählt werden, auch mit Hilfe weiterer Variablen. Verändern Sie z.B. Zeile 40 folgendermaßen:

```
40 N = N + (I*2)
```

Geben Sie RUN ein und Sie sehen, daß Ihr Zähler um einiges höher ist.

3.4 Zusammenfassung

Dieses Kapitel zeigte Ihnen die Leistungsfähigkeit Ihres Computers. Wir haben angefangen, richtig zu programmieren. Als eines der wichtigsten Konzepte haben Sie die "Variablen" kennengelernt. Das bedeutsamste Merkmal von Variablen ist, daß sie ihren Wert ändern, je nachdem, was Ihr Programm durchführt. Dies gilt nicht nur für numerische, sondern auch für Stringvariablen. Die verschiedenen Input-Kommandos haben Ihnen gezeigt, wie Sie Werten oder Texten Variablen zuweisen, abhängig davon, was Sie mit dem Programm durchführen. Als letztes haben wir die Schleifentechnik kennengelernt. Diese erlaubt Ihnen, dem Computer mit einem minimalen Aufwand mitzuteilen, daß er eine Reihe von Befehlen die durch zwei Anweisungen gekennzeichnet sind, mehrfach durchlaufen soll. Mit Schleifen können Sie Parameter für eine Operation mit gewünschter Schrittweise festsetzen, sich dabei zurücklehnen und Ihren COMMODORE-64 für sich arbeiten lassen.

Wie auch immer, Sie haben gerade erst zu programmieren gelernt. Im nächsten Kapitel werden Sie mehr Kommandos und Operationen kennenlernen, die Ihnen erlauben, tiefer in den Möglichkeiten des COMMODORE-64 zu forschen und damit die Programmierarbeiten zu vereinfachen. Je mehr Kommandos Sie kennen, desto einfacher wird es, ein Programm zu schreiben.

4

Verzweigungen im Programm

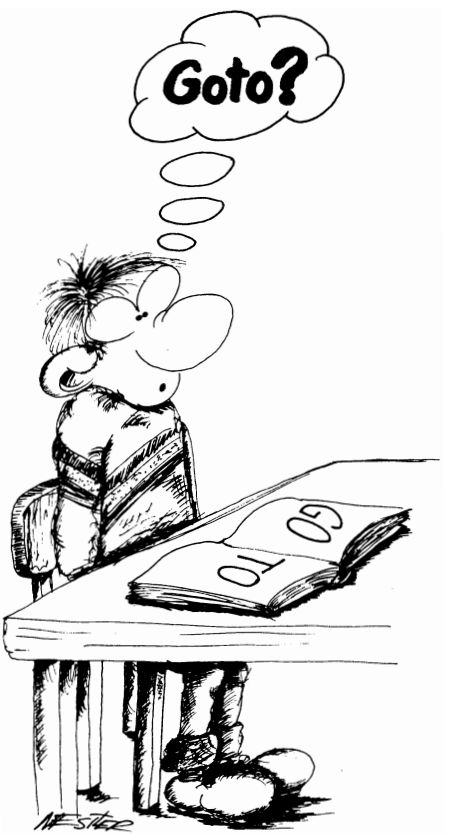
Einführung

In diesem Kapitel beginnen wir mit einer neuen Programmiertechnik, welche Ihre Fähigkeit, Programme zu schreiben, weiter erhöhen wird. Wir werden einige weiterentwickelte Techniken zeigen, und wenn Sie die Übungen Schritt für Schritt durchmachen, werden Sie die neuen Methoden beherrschen. Später, wenn Sie Ihre eigenen Programme entwerfen, sollten Sie den Mut haben, neue Kommandos auszuprobieren. Ein Problem haben alle Programmierer anfangs, nämlich, daß Sie nur die Kommandos verwenden, die Sie gelernt haben. Warum benutzen wir denn "kompliziertere" Kommandos, um das zu erreichen, was auch mit einfacheren Kommandos gemacht werden kann? Nun, die Antwort darauf ist sehr einfach. Welches Kommando ist denn nun wirklich einfacher, wenn ein "komplizierteres" Kommando die Arbeit von zehn "einfachen" Kommandos ausführen. Wenn Sie umfangreichere Programmanwendungen schreiben, werden Ihre Programme auch länger und neigen zu mehr Bugs. Je mehr Kommandos Sie dann zu prüfen haben, desto schwerer ist es, den Fehler zu finden. Während es deshalb in der Lernphase in Ordnung ist, lange Programme mit einer großen Anzahl einfacher Kommandos zu schreiben, sollten Sie später immer wieder daran denken, Programme durch den Gebrauch von komplexen Kommandos zu verkürzen.

Das Hauptanliegen ist, daß Sie Ihr Wissen über die verschiedenen Kommandos steigern und die Berechnungen vom Computer ausführen lassen. Dies mag Ihnen jetzt etwas seltsam erscheinen, aber Anfänger versuchen oft, alles für den Computer vorzukauen, und ihn als besseren Taschenrechner zu verwenden. Im letzten Kapitel - vielleicht erinnern Sie sich daran - haben wir einen Zähler angelegt, der die Anzahl der Schleifendurchläufe (mit der Schrittweite 3) protokolliert hat. Wir hätten auch selbst herausfinden können, wie oft die Schleife durchlaufen wurde, anstatt dies den Computer mit Hilfe des Counters (Zählers) ermitteln zu lassen. Damit hätten wir aber den Sinn des Programmierens verfehlt. Wenn Sie also neue Kommandos lernen, testen Sie gleich, wie Sie diese einsetzen können, damit das Programm Berechnungen ausführt, die Sie bisher selbst durchführen mußten.

4.1 Verzweigen

Bisher sind alle unsere Programme, einschließlich der Schleifen, von Anfang bis Ende linear durchlaufen worden. Ihr COMMODORE-64 kann jedoch auch einige Entscheidungen treffen. Dazu müssen wir ihm nur in der geeigneten Weise verschiedene Auswahlmöglichkeiten lassen. Wenn ein Programm den geraden Weg verläßt, wird dies entweder als Schleife oder als Verzweigung bezeichnet. Wir kennen bereits den Zweck einer Schleife, aber was ist eine Verzweigung? Wir wollen uns dies anhand der Kommandos IF/THEN und GOTO ansehen. Tatsächlich haben wir mit dem GET-Statement im letzten Kapitel diese Kommandos bereits eingeführt. Betrachten Sie folgendes Programm: (Hinweis: Sie sollten jetzt in der Lage sein, den Hauptspeicher mit NEW selbständig zu löschen, deshalb werde ich ab sofort dieses Kommando weglassen.)

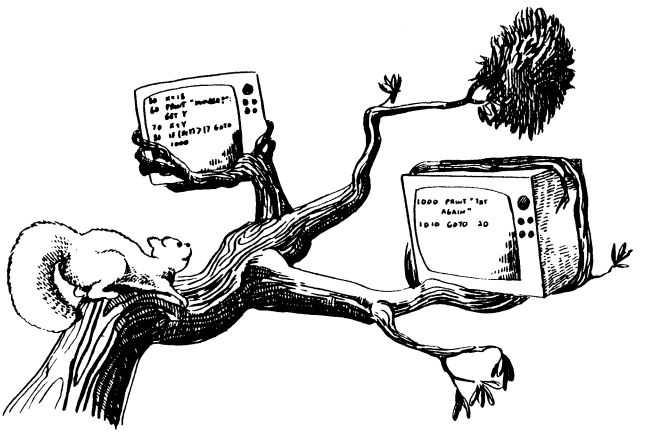


```
10 PRINT "<CLR/HOME>"
20 PRINT "WÄHLEN SIE BITTE ANHAND DER NUMMER AUS:"
30 PRINT
40 PRINT "1. BANANEN"
50 PRINT "2. ORANGEN"
60 PRINT "3. PFIRSICHE"
70 PRINT "4. WASSERMELONEN"
80 PRINT
90 INPUT "IHRE WAHL? ";X
100 PRINT "CLR/HOME"
110 IF X = 1 THEN GOTO 200
120 IF X = 2 THEN GOTO 300
130 IF X = 3 THEN GOTO 400
140 IF X = 4 THEN GOTO 500
150 GOTO 10:REM ES WIRD GEPRÜFT OB DER BEDIENER EINE DER
ZAHLEN 1 BIS 4 EINGEGEBEN HAT
200 PRINT "BANANEN":END
300 PRINT "ORANGEN":END
400 PRINT "PFIRSICHE":END
500 PRINT "WASSERMELONEN":END
```

Wie Sie sehen, "verzweigt" Ihr Computer zu der angegebenen Stelle, führt dort die Anweisungen durch und beendet das Programm. Das war nicht sehr eindrucksvoll, aber ein klares Beispiel. Versuchen wir jetzt ein praktischeres Beispiel, um Ihren Kindern die Mathematikaufgaben zu erleichtern.

```
10 PRINT "<CLR/HOME>"
20 AG$ = "ADDITIONS-SPIEL ":PRINT AG$
30 PRINT:PRINT
40 INPUT "BITTE ERSTE ZAHL EINGEBEN  -->";A
50 PRINT
60 INPUT "BITTE ZWEITE ZAHL EINGEBEN -->";B
70 PRINT
80 PRINT "WAS IST ";A;" + ";B;:INPUT C
90 IF C = A + B THEN GOTO 200
100 PRINT:PRINT "DAS IST LEIDER NICHT GANZ RICHTIG. BITTE
VERSUCHEN SIE ES NOCH EINMAL":PRINT
110 GOTO 80
200 PRINT "RICHTIG! SEHR GUT!"
210 PRINT
220 PRINT "WOLLEN SIE WEITERMACHEN? (J/N): ";
230 GET AN$:IF AN$ = "" THEN GOTO 230
240 IF AN$ = "J" THEN PRINT "<CLR/HOME>":GOTO 30
```

```
250 PRINT "<CLR/HOME>":PRINT:PRINT:PRINT
260 PRINT "SPIELEN SIE BALD WIEDER":END
```



Programm-Verzweigungen

Sie sehen, je mehr Kommandos Sie lernen, desto mehr Freude macht das Programmieren. Ändern Sie zum Spaß das Programm so ab, daß Sie Subtraktionen, Multiplikationen und Divisionen damit durchführen können.

Der Reiz liegt im Namen!

Kinder jedes Alters lieben es, ihren Namen auf dem Bildschirm zu sehen. Sehen Sie zu, ob Sie das Beispiel oben so abändern können, daß es zuerst den Namen des Kindes erfragt. Wenn das Programm dann die Antworten gibt, fügen Sie diese Variable mit ein. (z.B. DAS IST RICHTIG! SEHR GUT, MICHAEL). Benutzen Sie "NA$" als Variable für den Namen.

Sehen wir uns unser Programm genauer an, um etwas über die IF/THEN-Abfrage zu lernen. In Zeile 240 bewirkt die Verzweigung das Löschen des Bildschirms, wenn AN$ = "Y" ist. Bei einer anderen Antwort wird das Programm beendet. Sie werden sich fragen, warum das Programm nicht ohne Rücksicht auf die Antwort nach Zeile 30 verzweigt, wenn das Kommando "GOTO 30" nach einem Doppelpunkt steht und somit eine neue Zeile andeutet. Eine gute Frage. Der Grund dafür ist, daß das Programm nach einer IF-Abfrage sofort zu der nächsten Zeilennummer springt, wenn die Antwort oder Bedingung = Null ist. Das heißt, jede Anweisung nach einem Doppelpunkt in einer Zeile mit einer IF-Abfrage wird nur dann ausgeführt, wenn die Bedingung der IF-Abfrage zutrifft. Weiterhin wird die Bedingung von AN$ als "Y" abgefragt und nicht als einfaches Y ohne Anführungszeichen. Da der Benutzer aber ein Y und kein "Y" eingibt, nehmen wir an, daß das Programm das Y akzeptiert; denken Sie aber daran, daß AN$ ein "String" und keine numerische Variable ist. Deshalb müssen wir beim Setzen von Bedingungen darauf achten, welche Art von Variablen wir benützen. Wenn wir andererseits numerische Variablen, wie z.B. AN oder AN% verwenden, können wir die Abfrage so gestalten:

```
IF AN = 1 THEN ...
```

4.2 Vergleichsoperatoren

Bis jetzt haben wir nur "=" verwendet, um festzulegen, ob unser Programm verzweigen soll oder nicht. Es gibt jedoch noch weitere Bedingungen, sogenannte "Vergleichsoperatoren", die wir abfragen können. Sie sehen unten eine komplette Liste mit allen Vergleichsoperatoren, die wir verwenden können.

SYMBOL	BEDEUTUNG
=	gleich
<	kleiner als
>	größer als
<>	ungleich
>=	größer oder gleich
<=	kleiner oder gleich

Verzweigungen im Programm

Lassen Sie uns ein bißchen spielen und die Operationen auf ihre Leistungsfähigkeit überprüfen. Hier sind einige kleine Programme dafür:

```
10 PRINT "<CLR/HOME>"
20 INPUT "ZAHL 1 -->";A
30 INPUT "ZAHL 2 -->";B
40 IF A > B THEN GOTO 100
50 IF A < B THEN GOTO 200
60 IF A = B THEN GOTO 300
100 PRINT "ZAHL 1 IST GRÖßER ALS ZAHL 2":END
200 PRINT "ZAHL 1 IST KLEINER ALS ZAHL 2":END
300 PRINT "ZAHL 1 IST GENAUSO GROß WIE ZAHL 2"
```

```
10 PRINT "<CLR/HOME>"
20 "WOLLEN SIE WEITERMACHEN? (J/N)";AN$
30 IF AN$ <> "J" THEN END
40 GOTO 10
```

```
10 PRINT "<CLR/HOME>"
20 INPUT "WIE ALT SIND SIE? ";AG%
30 IF AG% >= 21 THEN GOTO 100
40 PRINT "<CLR/HOME>":PRINT:PRINT
50 PRINT "ES TUT MIR LEID, ABER FÜR DIESES SPIEL MÜSSEN SIE
ÄLTER ALS 21 SEIN!":END
100 PRINT "<CLR/HOME>":PRINT:PRINT "WAS MÖCHTEN SIE GERNE
TRINKEN?"
```

Nun, Sie wissen jetzt, wie Sie die Vergleichsoperationen in IF/THEN-Abfragen einsetzen können. Beachten Sie, daß diese sowohl mit Strings als auch mit numerischen Variablen funktionieren. Es gibt aber auch noch einen anderen Weg, Vergleichsoperatoren einzusetzen. Probieren Sie folgendes im Kommandomodus:

```
A = 10:B = 20:PRINT A = B
```

Ihr Computer antwortet mit 0, richtig? Es ist eine logische Operation. Wenn eine Bedingung falsch ist, antwortet Ihr COMMODORE-64 mit 0, wenn sie wahr ist, antwortet er mit -1. Versuchen Sie das folgende kleine Programm:

```
10 PRINT "<CLR/HOME>"
20 A = 10
30 B = 20
40 C = A > B
50 PRINT C
```

Wenn Sie nun RUN eingeben, erhalten Sie wieder die 0, da die Variable C wie folgt definiert war: A ist größer B. Weil jedoch A kleiner B war, ist die Variable falsch oder 0. Gehen wir einen Schritt weiter:

```
10 PRINT "<CLR/HOME>"
20 A = 10
30 B = 20
40 C = A > B
50 IF C = 0 THEN PRINT "A IST KLEINER ALS B":END
60 IF C = -1 THEN PRINT "A IST GRÖßER ALS B"
```

Später werden wir weitere Anwendungen für diese logischen Operationen auf dem COMMODORE-64 sehen. Im Augenblick ist es wichtig zu wissen, daß eine wahre Bedingung durch -1 und eine falsche Bedingung durch 0 dargestellt wird.

4.3 AND/OR/NOT

Manchmal müssen wir mehr als eine einfache Bedingung festsetzen. Stellen Sie sich vor, Sie möchten Ihre Finanzen in drei Kategorien einteilen: (1) unter DM 10, (2) zwischen DM 10 und DM 100 und (3) über DM 100. Mit unseren Vergleichsoperationen ist es einfach, die Eingaben unter DM 10 und über DM 100 zu unterscheiden. Was ist aber, wenn wir auch die Beträge dazwischen getrennt betrachten möchten? In diesem Fall hätten wir Schwierigkeiten ohne zusätzliche Kommandos. Die AND-, OR- und NOT-Bedingung erlaubt uns, mit unseren Vergleichsoperationen Bereiche zu setzen.

AND Wenn alle Aussagen wahr sind, ergibt AND wahr
OR Wenn eine Aussage wahr ist, ergibt OR wahr
NOT Wenn die Aussage nicht zutrifft, ergibt NOT wahr

Zum Beispiel:

```
10 PRINT "<CLR/HOME>"
20 INPUT "BITTE BETRAG EINGEBEN --> DM";A
30 IF A < 10 THEN 100
40 IF A > 10 AND A <= 100 THEN 200
50 IF A > 100 THEN 300
100 PRINT "WENIG ZUFRIEDENSTELLEND":GOTO 400
200 PRINT "GANZ ORDENTLICH":GOTO 400
300 PRINT "GROßARTIGES ERGEBNIS"
400 PRINT "WOLLEN SIE WEITERMACHEN? ";
410 GET AN$:IF AN$ = "" THEN 410
420 IF AN$ <> "J" AND AN$ <> "N" THEN PRINT "BITTE ANTWORTEN
SIE NUR MIT "J" ODER "N" ":GOTO 400
430 IF AN$ = "J" THEN 10
440 PRINT "<CLR/HOME>":PRINT "AUF WIEDERSEHEN"
```

In Zeile 40 haben wir die Bedingung gesetzt, daß das Programm bei größer 10 und kleiner oder gleich 100 verzweigen soll. Die Variable "A" hat beide Bedingungen zu erfüllen, damit verzweigt wird. Ebenso, in Zeile 420, haben wir die mit UND verknüpfte Bedingung dazu benutzt, um uns zu vergewissern, daß der Benutzer entweder "Y" oder "N" eingibt.

Wenn Sie das Programm sehr aufmerksam betrachtet haben, haben Sie sich bestimmt über einige eigenartige Formate gewundert. Da ste-

hen Zeilen mit IF/THEN-Abfragen, die nur mit THEN 100 oder ähnlichem fortgesetzt werden. Was ist passiert? Sollte da nicht ein GOTO stehen? Wieder haben wir eine neue Möglichkeit des COMMODORE-64-BASIC gefunden. Wenn Sie IF/THEN-Abfragen verwenden, ist es möglich, das GOTO wegzulassen und nur die Zeilennummer der Verzweigung anzugeben. Beachten Sie jedoch, daß wir auch GOTO-Anweisungen ohne Bedingungen im Programm haben: abgetrennt durch einen Doppelpunkt in der gleichen Zeile. Bis Sie mehr Übung im Programmieren haben, ist es besser, die GOTO-Anweisungen weiterhin auch bei den IF/THEN-Abfragen zu schreiben, auch wenn diese nicht erforderlich sind.

Sie haben vielleicht noch eine andere Frage, welche die AND-Bedingung in Zeile 420 betrifft. Normalerweise wenn wir sagen, etwas ist nicht "Y" oder "N", meinen wir, daß es das eine oder das andere sein muß. Beim Programmieren jedoch legen wir fest, daß das Programm verzweigt, wenn eine der Bedingungen zutrifft. So würde das Programm bei der Formulierung

```
420  IF  AN$ <> "J" OR AN$ <> "N" THEN PRINT "ANTWORTEN  SIE
BITTE MIT 'Y' ODER 'N'  ":GOTO 400
```

verzweigen, wenn AN$ ungleich "Y" oder ungleich "N" ist. Antworten wir jetzt zum Beispiel mit "Y", ist "Y" ungleich "N" und das Programm würde "ANTWORTEN SIE BITTE MIT 'Y' ODER 'N'" ausgeben, was wir nicht wollten. Um dies zu testen, ändern Sie in Zeile 420 das AND in ein OR und geben Sie RUN ein.

Probieren wir jetzt OR- und NOT-Operatoren in einem Programm:

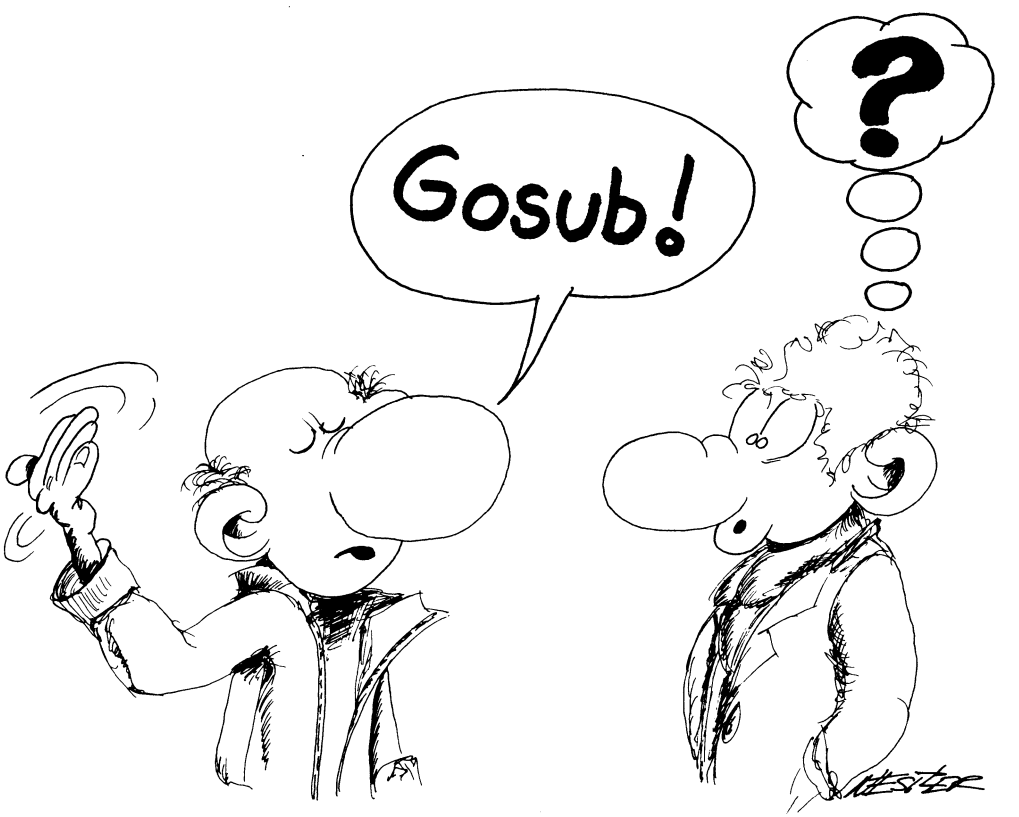
```
10 PRINT "<CLR/HOME>"
20 READ A
30 READ B
40 READ C
50 DATA 10,20,20
60 IF A + B = C OR A < B OR A - B = C THEN 100
70 END
100 PRINT "<CLR/HOME>"
110 PRINT "EINE DIESER AUSSAGEN MUSS RICHTIG SEIN"
```

In Zeile 60 sehen wir, daß A - B nicht gleich C, jedoch A + B gleich C, und A kleiner B ist. Bei der OR-Bedingung muß nur eine der Abfragen wahr sein, damit verzweigt werden kann. Probieren wir jetzt folgendes Programm:

Verzweigungen im Programm

```
10 PRINT "<CLR/HOME>"
20 READ A:READ B:READ C
30 DATA 10,20,30
40 Z = A - B = C
50 IF NOT Z THEN 100
60 END
100 PRINT "DAS IST RICHTIG! A - B = C IST FALSCH! - SAGTE
ICH DAS NICHT?"
```

Wie aus dem Beispiel ersichtlich ist, können wir die "Negation" verwenden, um eine Bedingung zu formulieren. In den meisten Fällen benutzen Sie <> (ungleich) oder die positive Abfrage, aber manchmal ist es einfacher, NOT einzusetzen.



4. 4 Unterroutinen

Sie haben in Ihrem Programm häufig gleiche Operationen, die sie an mehreren verschiedenen Stellen im Programm ausführen möchten. Sie können jetzt diese Reihe von Instruktionen immer wieder eingeben oder mit GOTOs zu Operationen springen und nach Ausführung mit GOTO wieder zurückkehren. Andererseits können Sie "Unterroutinen" (subroutines) schreiben, diese mit GOSUB ansteuern und nach Beendigung der Routine mit RETURN zum Ausgangspunkt zurückkehren. Im Prinzip arbeitet das GOSUB-Kommando genau wie das GOTO-Kommando - es schickt Ihr Programm zu einer Abfolge außerhalb des geradlinigen Programmablaufs. Auch durch das RETURN-Kommando schickt das Programm, wie das GOTO, zu einer Zeile außerhalb der normalen Reihenfolge. Wie auch immer, die GOSUB/RETURN-Folge ist einzigartig in der Arbeitsweise. Sehen wir uns dazu ein einfaches Beispiel an:

```
10 PRINT "<CLR/HOME>"
20 A$ = "HALLO":GOSUB 100
30 A$ = "WIE GEHT ES IHNEN HEUTE?":GOSUB 100
40 A$ = "MIR GEHT ES SEHR GUT":GOSUB 100
50 END
100 PRINT A$
110 RETURN
```

Unser Beispiel zeigt, daß eine Anweisung nach einem GOSUB ebenso abgearbeitet wird wie in der gleichen Zeile, nur daß sie in einem anderen Teil des Programms ausgeführt wird. Die RETURN-Anweisung bringt uns zu der nächsten Anweisung nach der GOSUB zurück. Die Verwendung von GOSUB/RETURN ist einfacher, da das RETURN Sie automatisch auf den Folgebefehl zurückbringt.

Um den Nutzen von GOSUB besser zu zeigen, verändern wir die Zeile 100. Probieren Sie folgendes:

```
100 L = LEN(A$)/2:PRINT TAB(20 - L);A$
```

Wenn Sie nun RUN eingeben, erscheinen alle Ihre Strings zentriert. Wie Sie sehen, dient diese kleine Routine dazu, Text zu zentrieren. Anstatt die Routine jedesmal, wenn Sie einen Text zentrieren möchten, neu zu schreiben, benutzen Sie nur das GOSUB zu der Zeile 100.

Sinnvolle Gliederung

Bis jetzt haben wir die Struktur des Programms nicht gesondert besprochen. Es war auch nicht notwendig. Wenn Ihre Instruktionsfolgen jedoch länger werden, wächst auch die Fehlerhäufigkeit. Wenn Sie bislang noch keinen Fehler gemacht haben, dann nur deshalb weil Sie sich streng an unsere Programme gehalten haben. Sie können, um speziell bei GOSUBs Fehler zu minimieren, das Programm in zusammenhängende "Blöcke" gliedern. Ein Block ist eine Unterroutine mit einer Anzahl von Zeilen. Zum Beispiel können Sie die Blöcke der Unterroutine in 100er oder 1000er gliedern, je nach der Länge der Blöcke. Das heißt, die Unterroutinen beginnen mit Zeilennummer 500, 600 und 700. Dabei ist es gleich, ob der Block 1 oder 10 Zeilen lang ist. Die Fehlersuche ist dadurch einfacher und es ist auch leichter für andere und Sie selbst nachzuvollziehen, was in dem Programm passiert. Es ist generell eine gute Programmiertechnik.

Errechnetes GOTO und GOSUB

Wir kommen jetzt zu einem seltsamen Beispiel, das während des Programmlaufs jedoch in einer klaren und einfachen Programmierung resultiert. Wie wir bereits sahen, können wir mit GOTO und GOSUB bei einer Bedingung verzweigen (z.B. IF A = 1 THEN GOTO 200). Ein einfacherer Weg, einen bedingten Sprung durchzuführen, ist die errechnete Verzweigung unter Verwendung der ON-Anweisung. Zum Beispiel:

```
10 PRINT "<CLR/HOME>"
20 INPUT "BITTE GEBEN SIE EINE ZAHL ZWISCHEN 1 UND 5 EIN ";A
30 IF A < 1 OR A > 5 THEN 20:REM PRÜFUNG
40 ON A GOSUB 100,200,300,400,500: REM ERRECHNETER GOSUB
50 PRINT "WOLLEN SIE FORTFAHREN? (J/N) ";
60 GET AN$:IF AN$ = "" THEN 60
70 IF AN$ <> "J" THEN END
80 GOTO 10
100 PRINT "EINS":RETURN
200 PRINT "ZWEI":RETURN
```

```
300 PRINT "DREI":RETURN
400 PRINT "VIER":RETURN
500 PRINT "FÜNF":RETURN
```

Um ein errechnetes GOSUB/GOTO durchzuführen, geben Sie eine Variable hinter der ON-Anweisung ein. Das Programm springt dann die Zeilennummern an, die durch Kommas getrennt sind. Wenn A = 1 ist, wird die erste Zeilennummer genommen, bei A = 2 die zweite, usw. Dies ist weit einfacher als die Eingabe: "IF A = 1 THEN GOSUB 100: IF A = 2 THEN GOSUB 200: usw.". Es ist jedoch notwendig, eine relativ kleine Ziffer für die "ON"-Variable zu nehmen, da die Anzahl der Unterroutinen begrenzt ist. Wenn Ihr Programm eine größere Zahl errechnet, wandeln Sie diese erst in kleinere Werte um. Zum Beispiel:

```
10 PRINT "<CLR/HOME>"
20 INPUT "GEBEN SIE IRGENDEINE ZAHL EIN -->";A
30 IF A < 100 THEN B = 1
40 IF A >= 100 AND A < 200 THEN B = 2
50 IF A >= 200 THEN B = 3
60 ON B GOSUB 100,200,300:REM AUFGRUND DER VARABLEN B
ERRECHNETER SPRUNG
70 PRINT "WEITER? (J/N) ";
80 GET AN$:IF AN$ = "" THEN 80
90 IF AN$ <> "J" THEN END
95 GOTO 10
100 PRINT "WENIGER ALS 100":RETURN
200 PRINT "MEHR ALS 100 ABER WENIGER ALS 200":RETURN
300 PRINT "MEHR ALS 200":RETURN
```

Lassen Sie das Programm ablaufen und geben Sie eine Nummer ein. Da das Programm nur in Abhängigkeit von der Variablen B, und nicht von A (der INPUT-Variablen), verzweigt, bekommen Sie keinen Fehler, da der größte Wert von B nur 3 erreichen kann.

Lassen Sie uns noch einmal zu den Vergleichsoperatoren zurückkehren, um zu sehen, wie diese in errechneten GOSUBs eingesetzt werden können. Erinnern Sie sich, daß die einzigen Zahlen, die wir bei Vergleichsoperationen erhalten, 0 oder 1 sind, was falsch oder wahr entspricht. Wir können die 0 oder 1 jedoch wie reguläre Zahlen benutzen. Probieren Sie folgendes:

```
10 PRINT "<CLR/HOME>"
20 X = 1:Y = 2: Z = 3
```

```
30 A = X < Z
40 B = Y > Z
50 C = Z > X
60 PRINT "A + A = ";A + A
70 PRINT:PRINT "A + B = ";A + B
80 PRINT:PRINT "A + B + C = ";A + B + C
90 END
```

Bevor Sie nun RUN eingeben, sollten Sie sich überlegen, was in Zeile 60, 70 und 80 ausgegeben wird. Haben Sie ein Ergebnis erhalten, geben Sie RUN ein und sehen Sie, was passiert. Probieren Sie es aus. Kommen Sie zurecht damit? Lassen Sie es uns Schritt für Schritt durchgehen:

1. Da X kleiner als Z ist, ist auch A "wahr" und erhält so den Wert eins (-1). Folglich ist A (-1 + -1) gleich -2.

2. Da Y nicht größer Z ist (Y = 2 und Z = 3), ist B falsch und erhält den Wert 0. Deshalb gibt A + B (-1 + 0) die Summe -1.

3. Da Z größer X ist, ist auch C "wahr" und erhält den Wert -1. Es ergibt sich daraus A + B + C (-1 + 0 + -1) gleich -2.

Sie haben es richtig gelöst? Gratuliere! Wenn nicht, Kopf hoch, und überarbeiten Sie es noch einmal. Erinnern Sie sich an die einfachen Ergebnisse und suchen Sie nicht nach komplizierten Erklärungen!

Sie wissen jetzt, wie Sie durch Verwendung von Vergleichsoperatoren Zahlenwerte erhalten und werden diese für errechnete GOSUBs benutzen. Das folgende Programm zeigt Ihnen wie:

```
10 PRINT "<CLR/HOME>"
20 INPUT "WIE GROSS WAR DIE ZUSCHAUERMENGE? ";HC
30 R = 1 + (HC >= 500) + (HC >= 1000)
40 IF R = 0 THEN R = 2
50 IF R = -1 THEN R = 3
60 ON R GOSUB 100,200,300
70 PRINT:INPUT "WEITER? (J/N) ";AN$
80 IF AN$ <> "J" THEN END
90 GOTO 10
100 PRINT "<CLR/HOME>"
```

```
110 PRINT "DIE ZUSCHAUER KAMEN NICHT SEHR ZAHLREICH - NICHT
EINMAL 500":RETURN
200 PRINT "<CLR/HOME>"
210 PRINT "EIN GUTER SCHNITT - IMMERHIN ZWISCHEN 500 UND
1000":RETURN
300 PRINT "<CLR/HOME>"
310 PRINT "DIE ZUSCHAUERMENGE WAR SEHR GROSS - 1000 ODER
MEHR":RETURN
```

Dieses Programm wird durch die Formel (Algorithmus) in Zeile 30 gesteuert. Lassen Sie uns diese genau ansehen:

1. Es gibt drei Bedingungen:

   a) HC ist kleiner als 500
   b) HC liegt zwischen 500 und 1000
   c) HC ist größer als 1000

2. Trifft die erste Bedingung zu, sind HC >= 500 und HC >= 1000 beide falsch. Also 1 + 0 + 0 = 1. Deshalb ist R = 1.

3. Wenn HC >= 500 aber kleiner 1000 ist, ist HC >= 500 wahr, aber HC >= 1000 falsch. Wir erhalten also 1 + (-1) + 0 = 0. Der Wert von R ist gleich 2.

4. Ist HC >= 500 und >= 1000, ergibt sich mit unserer Formel 1 + (-1) + (-1) = -1. R erhält den Wert 3.

Eine kleine Pause

Lassen Sie uns hier eine kleine Pause für einen Rückblick einlegen. Beim Programmieren gibt es keinen RICHTIGEN oder FALSCHEN Weg. Manche Programme sind nur leistungsfähiger, schneller oder brauchen weniger Speicherplatz als andere. Wenn ein Programm das ausführt, was Sie wollten, spielt es keine Rolle, wie langsam es ist oder wie lange Sie zur Erstellung brauchten: es ist "gut". Im Beispiel oben haben wir einen Algorithmus mit Vergleichsoperatoren benutzt, um das zu erreichen, was wir sonst nur mit mehr Codieraufwand erreichen würden. Arbeiten Sie mit solchen Formeln aber erst, wenn Sie sich das nötige Grundwissen in Mathematik angeeignet haben. Wenn Sie nicht häufig mit Algorithmen arbeiten, erwarten Sie nicht, sofort die volle Wirkungsweise zu verstehen. Der Algorithmus, den wir benutzt haben, ist relativ einfach. Wenn Sie sich weitere Programme ansehen, werden Sie auch etwas kompliziertere entdecken. Die Hauptsache ist, am Ball zu bleiben. Durch die Praxis werden Sie alle Formeln und Programmkürzungen lernen. Denken Sie daran: solange Sie Ihre Programme so zum Laufen bringen, wie Sie sich das vorgestellt haben, sind Sie auf dem "richtigen Weg".

Strings und Vergleichsoperatoren

Bevor wir den Abschnitt der errechneten GOTOs und GOSUBs mit Vergleichsoperatoren verlassen, sehen wir uns noch an, wie diese auf Strings einwirken. Probieren Sie folgendes aus:

```
A$ = "A":B$ = "B":PRINT B$ > A$ <RETURN>
```

Nun, überrascht? Vergleichsoperatoren können also nicht nur numerische Variablen, sondern auch alphabetische Strings vergleichen, wobei "A" den kleinsten und "Z" den größten Wert darstellt. Bei der Frage, ob B$ größer als A$ ist, erhalten wir "-1" (wahr), da B$ ein B und A$ ein A enthält. Sie fragen sich jetzt vielleicht, was Sie mit diesem neuen Wissen anfangen können. Nun, beim Sor-

tieren von Strings (z.B. eingegebene Namen in die alphabetische Reihenfolge bringen) ist diese Operation hilfreich. Wir zeigen Ihnen später eine Routine, mit der Sie Strings sortieren können. Zunächst aber noch ein kleines Beispiel zum Sortieren von zwei eingegebenen Strings.

```
10 PRINT "<CLR/HOME>"
20 INPUT "WORT #1 -->";A$
30 INPUT "WORT #2 -->";B$
40 PRINT:PRINT:PRINT
50 IF A$ < B$ THEN PRINT A$:PRINT B$
60 IF A$ > B$ THEN PRINT B$:PRINT A$
```

Das war's! Ein Programm, das zwei Wörter in eine alphabetische Reihenfolge bringt.

4.5 Tabellen

Am besten ist, Sie stellen sich unter einer Tabelle eine Art Variable vor. Sie wissen bereits, daß Sie Namen wie A, D$, KK% oder X1 an Variablen vergeben können. Eine Tabelle benutzt einen einzigen Namen, um verschiedene Variablen zu unterscheiden. Betrachten Sie die beiden folgenden Listen. Die eine enthält normale Variablen, die zweite benutzt eine String-Tabelle:

STRING VARIABLE	**STRING TABELLE**
S$ = "KATZE"	AM$(1) = "KATZE"
H$ = "HUHN"	AM$(2) = "HUHN"
D$ = "DOGGE"	AM$(3) = "DOGGE"
P$ = "PFERD"	AM$(4) = "PFERD"

Wenn Sie jetzt D$ ausgeben, sehen Sie DOGGE, ebenfalls bei der Ausgabe von AM$(3). Auch für numerische Variablen können wir Tabellen definieren, z.B.

```
A(1) = 1
A(2) = 2
A(3) = 3
A(4) = 4  usw.
```

Sie werden sich erneut fragen: "Warum nehmen wir nicht normale Strings oder numerische Variablen anstelle der Tabelle?" Erstens: ist es bei Benutzung einer Tabelle einfacher, den Überblick darüber zu behalten, was im Programm abläuft. Zweitens spart eine Tabelle jede Menge Zeit. Sehen Sie sich das folgende Programm zur Eingabe von 10 Namen an, das eine String-Tabelle benutzt:

```
10 PRINT "<CLR/HOME>"
20 FOR I = 1 TO 10
30 PRINT "NAME #";I;:INPUT NA$(I)
40 NEXT I
50 FOR I=1 TO 10:PRINT NA$(I)
60 NEXT I
```

Schreiben Sie nun ein Programm mit normalen Strings, das die gleiche Verarbeitung durchführt. Es wird sehr viel mehr Codieraufwand nötig sein, aber versuchen Sie es ruhig. Benutzen Sie dazu die Variablen N0$ bis N9$.

Nach der Erstellung dieses Programms sehen Sie, wieviel Zeit es spart, Tabellen zu verwenden. Bevor wir weitermachen, sehen wir uns noch genau an, wie das Programm mit der FOR/NEXT-Schleife und der Tabelle arbeitet:

1. Die FOR/NEXT-Schleife erzeugt fortlaufende Nummern, so daß folgende Tabelle entsteht:

```
FOR I = 1 TO 10
NA$(1) <-- ERSTER SCHLEIFENDURCHLAUF
NA$(2) <-- ZWEITER SCHLEIFENDURCHLAUF
NA$(3) <-- DRITTER SCHLEIFENDURCHLAUF
NA$(4) usw.
NA$(5)
NA$(6)
NA$(7)
NA$(8)
NA$(9)
NA$(10)
NEXT I
```

2. Jede Stringvariable des Benutzers wurde in der fortlaufend numerierten Tabellenvariablen gespeichert.

3. Die Ausgabe durch PRINT wurde mit der FOR/NEXT-Schleife durch Wiedergabe der Nummern in die Tabellenvariable programmiert.

Um sich mit der Idee anzufreunden, daß die Tabellenvariable auch nur ein Variablentyp ist, geben Sie folgendes ein:

```
A(10) = 432:PRINT A(10) <RETURN>
XYZ(9) = 2.432:PRINT XYZ(9) <RETURN>
R2D2$(1) = "BEEP!" + CHR$(7):PRINT R2D2$(1) <RETURN>
J%(5) = 321:PRINT J%(5) <RETURN>
```

Nun, Sie denken jetzt vielleicht beim Wort "Tabellen" nicht mehr an etwas Exotisches.

Das DIMensionieren von Tabellen

Wenn Sie sehr aufmerksam mitgearbeitet haben, ist Ihnen sicher aufgefallen, daß wir keine größere Zahl als 10 verwendet haben. Der Grund dafür ist, daß Sie durch die DIM-Anweisung Speicherplatz für die Tabelle reservieren müssen, wenn diese größer als 10, genauer gesagt: größer als 11 ist (0 - 10 entspricht 11 Tabellenelementen). Sie sehen unten ein Beispiel mit einem Format der DIM-Anweisung.

```
10 PRINT "<CLR/HOME>"
20 DIM AB(150):REM DIMENSIONIERUNG FÜR DIE TABELLE
30 FOR I = 1 TO 150
40 AB(I) = I
50 NEXT I
60 FOR I = 1 TO 150
70 PRINT AB(I),
80 NEXT I
```

Geben Sie das Programm und RUN ein. (Das Programm müßte laufen.) Löschen Sie nun Zeile 20 durch Eingabe der Zeilennummer. Wenn Sie jetzt RUN eingeben, bekommen Sie einen Fehler für die nicht dimensionierte Tabelle (? BAD SUBSCRIPT ERROR IN 40 - da sich in

Zeile 20 jetzt keine DIM-Anweisung mehr befindet). Achten Sie also bei Tabellen, die mehr als 11 Elemente (von 0 bis 10) enthalten, auf die DIM-Anweisung.

Besser Sicherheit als Probleme

Viele Programmierer dimensionieren alle Tabellen, ohne Rücksicht auf die Anzahl der Elemente in der Tabelle. Dies ist durchaus richtig und Angaben wie DIM X$(3) oder DIM N%(5) sind gültig. Wenn Sie Programme aus Zeitschriften und Büchern übernehmen, werden Sie häufig diese niedrigen DIM-Anweisungen vorfinden. Nicht weil sie notwendig sind, sondern weil die Programmierer es einem klaren Programmstil zuliebe für gut halten, alle Tabellen zu dimensionieren.

Mehrdimensionale Tabellen

Bis jetzt haben Sie nur eindimensionale Tabellen kennengelernt. Es ist aber auch möglich, Tabellen mit zwei oder mehr Dimensionen anzulegen. Beginnen wir mit zweidimensionalen Tabellen und schauen wir, wie Tabellen mit mehr als einer Dimension anzuwenden sind.

Am besten stellen Sie sich eine zweidimensionale Tabelle als Matrix vor. Zum Beispiel enthält folgende Reihe von 1 bis 3, eine Tabelle in 2 Dimensionen: A(1,1) A(1,2) A(1,3) A(2,1) A(2,2) A(2,3) A(3,1) A(3,2) A(3,3). Bei der Einteilung in eine Matrix können wir die erste Ziffer als Zeilen- und die zweite als Spaltenindex betrachten. Das Beispiel macht es etwas klarer:

	SPALTE #1	SPALTE #2	SPALTE #3
ZEILE #1	A(1,1)	A(1,2)	A(1,3)
ZEILE #2	A(2,1)	A(2,2)	A(2,3)
ZEILE #3	A(3,1)	A(3,2)	A(3,3)

Es ist wichtig, sich daran zu erinnern, daß jedes Element der Tabelle nur einen Variablentyp darstellt. Um sich dies einzuprägen, probieren Sie folgendes aus:

```
XV$(3,1) = "ICH BIN EINE VARABLE":PRINT XV$(3,1) <RETURN>
JK%(2,2) = 21:PRINT JK% <RETURN>
MM(1,1) = 3.212:PRINT MM(1,1) <RETURN>
```

Um Tabellen mit all ihren Vorteilen im Programm einzusetzen, müssen sie als Variablen betrachtet werden. Im nächsten Programm, das die 12 Mitglieder eines Orchesters in einer Matrix auflistet, benutzen wir eine zweidimensionale Tabelle.

```
10 PRINT ">CLR/HOME>"
20 DIM BA$(4,4):REM FÜR 3 SPALTEN UND 3 ZEILEN
30 FOR I = 1 TO 4:REM ZEILEN
40 FOR J=1 TO 4:REM SPALTEN
50 READ BA$(I,J)
60 NEXT J
70 NEXT I
80 DATA RALF,PETER,MARLENE,FRANK,HANS,DAVID,KARL,UWE
90 DATA ERICH, MARIA, TOM, SUE, PAUL, JOSEF, NANA, BETTINA
100 REM AUSGABEBLOCK
110 FOR I=1 TO 4:REM ZEILEN
120 FOR J=1 TO 4:REM SPALTEN
130 PRINT BA$(I,J),:REM KOMMA WIRKT ALS TABULATOR AUF JEDE
14. STELLE
140 NEXT J
150 NEXT I
```

Wenn Sie das Programm ablaufen lassen, wird jedes der Mitglieder aufgelistet. Sie können das gleiche auch mit einer Dimension ausführen, da Sie die PRINT-Ausgabe in Zeile 130 auch mit Komma formatieren könnten. Was ist also besser an der zweidimensionalen Tabelle? Um dies zu sehen, erweitern wir unser Programm um einige Zeilen:

```
160 PRINT:PRINT "ZUM FORTFAHREN IRGENDEINE TASTE BETÄTIGEN";
170 GET AN$:IF AN$ = "" THEN 170
180 PRINT "<CLR/HOME>"
185 PRINT "WELCHE ZEILE & SPALTE MÖCHTEN SIE SEHEN?"
190 INPUT "ZEILE # ->";R
200 INPUT "SPALTE #->";C
210 PRINT:PRINT BA$(R,C);" IST DER INHALT VON ZEILE ";R;
```

```
" SPALTE ";C
220 PRINT:PRINT "WEITERE ANZEIGEN? (J/N) ";
230 GET M$:IF M$ = "" THEN 230
240 IF M$ ="J" THEN 180
```

Sie können jetzt den Wert oder Inhalt eines speziellen Elements in zwei Dimensionen lokalisieren. Wenn Sie in unserem Beispiel die Zeilen- und Spaltennummer wissen, finden Sie das Mitglied dieser Position. Die Verwendung von zweidimensionalen Tabellen ist in Fällen, die mit einer Matrix bearbeitet werden, eine wichtige Ergänzung Ihrer Programmierkommandos.

Es ist auch möglich, mehrere Dimensionen in einer Tabellenvariablen zu haben. Wenn Sie mit mehreren Dimensionen arbeiten, dürfen Sie die verschiedenen Aspekte einer eindimensionalen Tabelle nicht außer acht lassen. Manchmal, wenn eine mehrdimensionale Tabelle schwer zu definieren oder zu handhaben ist, ist es besser, sie in mehrere ein- oder zweidimensionale Tabellen zu teilen. Lassen Sie uns ein Beispiel mit einer dreidimensionalen Tabelle durchgehen (Übrigens basiert dieses Problem auf einer wirklichen Anwendung):

```
10 PRINT "<CLR/HOME>"
20 PRINT "WEINKELLER-ORGANISATION"
30 PRINT:PRINT "WIEVIELE REGALE, ZEILEN, SPALTEN?"
35 INPUT "(BITTE WERTE DURCH KOMMA GETRENNT EINGEBEN);RK,R,C
40 DIM WI$(RK,R,C)
50 INPUT "WIEVIELE FLASCHEN WOLLEN SIE UNTERBRINGEN? ";N%
60 PRINT:FOR I = 1 TO N%
70 INPUT "REGAL #->";RA
80 INPUT "ZEILE #->";RO
90 INPUT "SPALTE #->";CO
100 INPUT "NAME DES WEINES: ";WN$
110 WI$(RA,RO,CO) = WN$
120 NEXT I
200 REM ROUTINE FÜR DIE ÜBERPRÜFUNG DES WEINKELLERS
210 PRINT "<CLR/HOME>"
215 INPUT "WELCHES REGAL WOLLEN SIE ÜBERPRÜFEN? #-> ";RR
220 FOR I = 1 TO R
230 FOR J = 1 TO C
240 IF WI$(RR,I,J) = "" THEN WI$(RR,I,J) = "LEER"
250 PRINT "REGAL #";RR;"ZEILE #";I;"SPALTE #";J;"ENTHÄLT";
```

```
WI$(RR,I,J)
260 NEXT J
270 NEXT I
280 END
```

Dies war ein relativ langes Programm. Sehen Sie es genau durch, damit Sie verstehen, was es macht. Lassen Sie mich nochmals daran erinnern, daß eine dreidimensionale Tabelle nur eine Variable mit einer Menge Ziffern in Klammern darstellt. Achten Sie auch in Zeile 35 darauf, wie wir mit einer einzigen INPUT-Anweisung mehrere Werte eingeben. Wir benutzen das Format

```
INPUT A,B,C
```

und solange dem Operator (Benutzer) mitgeteilt wird, wie viele Eingaben, durch Komma getrennt, er zu machen hat, ist alles in Ordnung. Es wäre gut, dieses Programm als Beispiel für eine mehrdimensionale Tabelle auf Diskette zu speichern.

4.6 Zusammenfassung

In diesem Kapitel haben Sie sehr viel gelernt. Wenn Sie alles verstanden haben - sehr gut! Wenn nicht, lassen Sie den Kopf nicht hängen. Wenn Sie mehr Praxis haben, wird alles viel klarer. Probieren Sie auf alle Fälle die Beispiele aus. Seien Sie mutig, und wagen Sie Experimente mit den erlernten Kommandos. Solange Sie eine Übungsdiskette besitzen, löschen Sie schlimmstenfalls einige Programme.

Sie haben gelernt, daß Ihr COMMODORE-64 rechnen kann. Mit der IF/THEN-Abfrage und den Vergleichsoperatoren geben Sie Ihrem Computer die Fähigkeit der "Entscheidungsfindung". Durch die Unterroutinen ist es möglich, an Entscheidungspunkten in einen anderen Teil des Programms zu verzweigen. Errechnete GOTOs und GOSUBs erlauben die Durchführung entsprechender Sprünge mit minimalem Programmieraufwand.

Zuletzt haben wir uns die Tabellenvariablen genauer angesehen. Tabellen erlauben uns, Werte in fortlaufend angeordneten Variablen oder Elementen einzugeben. Durch FOR/NEXT-Schleifen ist es möglich, schnell mehrfach vorkommende Variablen gleicher Art, bis zur Grenze der festgelegten DIMension, zu bearbeiten. Die Tabellen nützen uns nicht nur bei der Verarbeitung der geordneten Variablen, sondern ersparen uns auch eine Menge Zeit.

Im nächsten Kapitel beginnen wir mit Variablen zu arbeiten, die uns helfen, alles zu arrangieren. Da unsere Programme jetzt immer länger und komplizierter werden, müssen wir mit großer Sorgfalt arbeiten. Wir können besser klare und nützliche Programme erstellen, indem wir sie in kleine, handliche Abschnitte gliedern.

5

Organisieren von Programmteilen

Einführung

Wenn Sie immer mehr Informationen eingeben und diese nicht organisieren, werden die Programme verwirrend. Eine gute Organisation erlaubt Ihnen, komplexere und längere Programme zu verarbeiten. Diese Organisation kommt mit der Erfahrung im Programmieren von selbst. Beherrschen Sie mehr Kommandos, können Sie mehr machen. Aber je mehr Sie machen, desto größer wird die Gefahr, sich im Programm zu verlieren.

Ausgaben zu formatieren ist einer der wichtigsten Punkte, der dabei hilft, den "Overflow" (Überlauf) zu verhindern. Variablen mischen sich, Tabellen sind falsch numeriert und der Bildschirm ist ein Chaos. Um diese Probleme in den Griff zu bekommen, werden wir uns intensiv mit der Formatierung von Texten und Strings beschäftigen. Nicht nur, damit wir in der Lage sind, Ausgaben zu steuern, sondern auch wegen des Programmstils.

Der zweite Hauptbereich in bezug auf die Unordnung ist der I/O (INPUT/OUTPUT). Einige der Probleme in diesem Bereich hängen mit der Formatierung zusammen; grundlegender ist jedoch das Problem der Organisation der Ein- und Ausgabe. Daten müssen mit den passenden Variablen verbunden und zur richtigen Verarbeitung bereitgestellt werden. Zusätzlich werden wir uns die Organisation der Datenverarbeitung ansehen.

5.1 Formatieren von Text

In Kapitel 1 erwähnten wir, daß die Tastatur des COMMODORE-64 der einer Schreibmaschine ähnlich ist. Bei einer Schreibmaschine können Sie Tabulatoren setzen, so daß der Text einen festgelegten Leerraum am linken Rand aufweist. Beim COMMODORE-64 haben Sie eine TAB und SPC Funktion, deren Bedeutung wir uns gleich ansehen:

KOMMANDO	BEDEUTUNG
TAB(n)	wird in der PRINT Anweisung benutzt, um das nächste Zeichen n Spalten vom linken Rand zu plazieren.
SPC(n)	wird in der PRINT Anweisung benutzt und stellt dem String n Leerzeichen voran.
<HOME>	plaziert den Cursor in der linken oberen Ecke des Bildschirms. Wird ohne Drücken der SHIFT Taste betätigt.

Um besser zu verstehen, wie diese Kommandos Text formatieren, wollen wir sie gleich einsetzen.

```
10 PRINT "<CLR/HOME>":PRINT:PRINT
20 PRINT TAB(20);"HIER WURDE DER TABULATOR GESETZT"
30 PRINT SPC(20);"BIS HIERHER WURDEN LEERZEICHEN EINGEFÜGT"
40  PRINT "<HOME>";"HIER IST OBEN":REM DRÜCKEN SIE DIE TASTE
<CLR/HOME> OHNE DIE SHIFT-TASTE - SIE BEKOMMEN EIN INVERSES
"S"
50 FOR I = 1 TO 20:PRINT:NEXT:PRINT "HIER IST UNTEN"
```

Achten Sie nach Eingabe von RUN darauf, daß die Benutzung von <HOME> nicht den Bildschirm löscht. Das Kommando setzt den Cursor nur in die linke obere Ecke des Monitors und läßt die Ausgabe von Zeile 20 und 30 auf dem Bildschirm stehen. Sie sind auch in der Lage, ein vertikales TAB zu erzeugen, indem Sie eine leere PRINT-Anweisung in Zeile 50 dazu benutzen, den Text in die vertikale Position 20 auf den Bildschirm zu bringen. Der restliche Text auf dem Bildschirm wurde dabei nicht gelöscht. Lassen Sie uns jetzt ein bißchen mit den Kommandos spielen. Hier ist ein kleines Programm, das Ihnen zeigt, wie Sie im Programm Text plazieren können.

```
10 PRINT "<CLR/HOME>":FOR I = 1 TO 4:PRINT:NEXT
20 INPUT "BITTE NACHRICHT EINGEBEN -->";MS$
30 PRINT:INPUT "HORIZONTALER PLATZ (1-40) ->";H
40 PRINT:INPUT "VERTIKALER PLATZ (1-25) ->";V
50 PRINT "<CLR/HOME>"
60 FOR VER = 1 TO V:PRINT:NEXT VER:PRINT TAB(H);MS$
70 PRINT:PRINT "UM FORTZUFAHREN, IRGENDEINE TASTE DRÜCKEN"
75 PRINT "ZUM BEENDEN E-TASTE DRÜCKEN     ";
80 GET A$:IF A$ = "" THEN 80
90 IF A$ <> "E" THEN 10
100 END
```

Wie Sie sehen, können Sie zur Textformatierung auch Variablen verwenden. TAB(H) wird gelesen wie TAB(10) oder TAB(15) oder TAB einer beliebigen anderen Ziffer zwischen 1 und 40. Was, denken Sie, passiert in dem oben gezeigten Programm, wenn Sie einen String an der 39sten horizontalen und der 25sten vertikalen Position eingeben? Da der maximale TAB 40 ist und die maximale vertikale Position 25, wird der Text über den Rand hinausgehen. Probieren Sie es selbst aus, um zu sehen, was passiert. Es wäre gut, die Grenzen von TAB und der vertikalen Position mit diesem Programm zu testen, um Klarheit über die Verwendung dieser Parameter zu bekommen.

5.2 Aufteilen von Strings

Bis jetzt haben wir nur "ganze" Strings besprochen. Ohne Rücksicht darauf, wie kurz oder lang diese Strings sind, bezeichnen wir sie als "ganze" Strings. Wenn wir zum Beispiel R$ als "GEHEN" definieren, können wir "GEHEN" als das gesamte R$ bezeichnen. Ebenso: wenn wir "DIES IST EIN LANGER STRING" ALS R$ definieren, ist "DIES IST EIN LANGER STRING" das gesamte R$. Hier ergibt sich die Möglichkeit, Teile des Strings einzeln zu benutzen oder verschiedene Strings zu einem zusammenzufassen. Wenn wir zu Datenbankprogrammen kommen, werden wir dies als sehr notwendig erkennen. Es gibt auch Anwendungen, bei denen wir die Länge eines Strings benötigen, einen numerischen Wert des Strings finden müssen und sogar Strings in numerische Variablen, und wieder zurück, umwandeln müssen.

Glauben Sie mir!

Ich gebe es nicht gerne zu, aber als ich zum ersten Mal von den Kommandos, die wir hier behandeln, hörte, dachte ich mir: "Welch eine Zeitverschwendung!" Es ist genug, das einfache Material durchzugehen, warum möchte da jemand auch noch Strings teilen und wieder zusammenfügen? Wenn man nur einen bestimmten Teil eines Strings benötigt, warum diesen nicht einzeln definieren? Und wenn man einen längeren String braucht, dann definiert man diesen eben einfach länger! Das waren meine Gedanken über die Stringformatierung. Ich bin jedoch zu der Überzeugung gelangt, daß ich es für sehr schwierig halte, mir ein Programm ohne diese leistungsfähigen Kommandos auch nur vorzustellen. Glauben sie mir! Kommandos zur Stringformatierung sind sehr wichtig und wenn Sie dies im Moment vielleicht noch nicht erkennen, werden Sie es sehen, wenn Sie bereits mehr Programme erstellt haben.

Formatieren von Strings

Wir werden den Abschnitt, Formatieren von Strings, in vier Teilen besprechen: 1) Errechnen der Länge eines Strings, 2) Finden von Teilen eines Strings, 3) Umwandeln von Text in Zahlen und wieder zurück und 4) Verbindung von Strings (Verkettung).

Errechnen der Länge eines Strings

Manchmal ist es wichtig, die Länge eines Strings zur Druckausgabe zu berechnen. Glücklicherweise besitzt Ihr COMMODORE-64 eine gute Möglichkeit, Ihnen die Länge eines einzelnen Strings mitzuteilen. Durch das Kommando PRINT LEN(A$), erhalten Sie die Anzahl der Zeichen, inclusive Leerstellen, die Ihr String enthält. Probieren Sie das folgende kleine Programm, um zu sehen, wie das Kommando arbeitet:

```
10 PRINT "<CLR/HOME>"
20 INPUT "BITTE TEXT EINGEBEN ->";A$
30 PRINT A$;" HAT";LEN(A$);" BUCHSTABEN"
40 PRINT:PRINT "WEITER? (J/N) ";
50 GET AN$:IF AN$ = "" THEN 50
60 IF AN$ = "J" THEN 20
```

Um eine weitere Anwendung kennenzulernen, sehen wir uns eine abgeänderte Version der Zentrierroutine aus dem letzten Kapitel an:

```
10 PRINT "<CLR/HOME>"
20 PRINT "GEBEN SIE BITTE EINEN TEXT MIT BIS ZU 40 ZEICHEN
EIN":INPUT "-> ";S$
30 PRINT "<CLR/HOME>"
40 L = 20 - LEN(S$) / 2:PRINT TAB(L);S$
50 FOR I=1 TO 20:PRINT:NEXT
55 PRINT "WEITER MIT BELIEBIGER EINGABE - ENDE = E-TASTE";
60 GET A$:IF A$ = "" THEN 60
70 IF A$ <> "E" THEN 10
80 END
```

Wir können jetzt versuchen, die Eingabe mit dem LEN-Kommando zu steuern, um zu sehen, wie wir die Länge eines Strings errechnen und damit die TAB-Anweisung beeinflussen. Stellen Sie sich vor, Sie möchten ein Programm schreiben, das Adreßaufkleber druckt, die alle nur 30 Zeichen lang sind. Vorher möchten Sie sich vergewissern, daß Ihre Eingaben 30 Zeichen (incl. Leerzeichen) nicht überschreiten. Wir schreiben dazu ein Programm, das die Länge eines Strings prüft, bevor dieser akzeptiert wird.

```
10 PRINT "<CLR/HOME>"
20 PRINT "BITTE EINEN NAMEN MIT MAX. 30 STELLEN EINGEBEN -
LEERSTELLEN ZÄHLEN MIT"
30 INPUT "KEINE KOMMAS VERWENDEN ->";NA$
40 IF LEN(NA$) > 30 THEN GOTO 100:REM PRÜFUNG
50 PRINT:PRINT NA$
60 PRINT:PRINT "WEITEREN NAMEN? (J/N) ";
70 GET AN$:IF AN$ = "" THEN 70
80 IF AN$ <> "J" THEN END
90 GOTO 10
100 PRINT "<CLR/HOME>"
110 PRINT "SIE HABEN MEHR ALS 30 ZEICHEN EINGEGEBEN"
120 PRINT:GOTO 20
```

Verstoßen Sie nun gegen die Anweisung in Zeile 20. Geben Sie einen String ein, der länger als 30 Zeichen ist und sehen Sie was passiert. (Wenn Ihr Computer das nicht mag, können Sie das Programm jederzeit neu starten. Dies sollte Sie nur daran erinnern.) Wurde das Programm richtig übernommen, bekommen Sie bei der Eingabe von mehr als 30 Zeichen pro String eine Meldung vom Programm.

In den Beispielen oben sehen Sie, wie das LEN-Kommando verschieden eingesetzt werden kann. Es gibt noch viele andere Anwendungen dieses Kommandos z.B. die zur Reduzierung der Laufzeit des Programms dienen und die Informationen berechnen. Um den Nutzen dieser Kommandos zu verstehen, muß man damit experimentieren und beobachten, wie andere Programmierer sie einsetzen.

Finden der mittleren (MID$), linken (LEFT$) und rechten (RIGHT$) Teile eines Strings

Stellen Sie sich vor, Sie möchten eine einzige Variable benutzen, um drei verschiedene Zustände, wie z.B. SCHLECHT, GUT, AUSGEZEICHNET, zu beschreiben, aber nur Teile davon ausgeben. Durch Verwendung von MID$, LEFT$ und RIGHT$ ist es möglich, nur den Teil des Strings auszugeben, den Sie brauchen. Zum Beispiel benutzen wir im nächsten Programm einen String, um drei verschiedene Zustände darzustellen:

```
10 PRINT "<CLR/HOME>"
20 X$ = "SCHLECHT GUT AUSGEZEICHNET"
30 PRINT "WIE GEHT ES IHNEN HEUTE? (<S>CHLECHT, <G>UT ODER
<A>USGEZEICHNET) ";
40 GET F$:IF F$ = "" THEN 40
50 IF F$ = "S" THEN PRINT LEFT$(X$,8)
60 IF F$ = "G" THEN PRINT MID$(X$,10,3)
70 IF F$ = "A" THEN PRINT RIGHT$(X$,13)
80 PRINT:PRINT:PRINT "NOCHMAL? (J/N) ";
90 GET AN$: IF AN$ = "" THEN 90
100 IF AN$ = "J" THEN 10
```

Lassen Sie uns festhalten, daß es leichter wäre, einfach zur Ausgabe von 'SCHLECHT','GUT' und 'AUSGEZEICHNET' zu verzweigen, und es wäre ebenso effizient. Wir wollten jedoch nur der Übung halber

dieses Beispiel machen und nicht, um die Programmorganisation zu optimieren. Lassen Sie uns sehen, was die neuen Kommandos durchführen.

KOMMANDO	**BEDEUTUNG**
MID$(A$,N,L)	Findet den Teil von A$, beginnend beim Zeichen N, in der Länge von L Zeichen.
LEFT$(A$,L)	Findet den Teil von A$ L Zeichen lang, beginnend an der linken Seite des Strings.
RIGHT$(A$,L)	Findet den Teil von A$ L Zeichen lang, beginnend an der rechten Seite des Strings.

Um Ihnen sofort etwas Erfahrung mit diesen neuen Kommandos zu ermöglichen, probieren Sie folgendes:

```
W$ = "WAS FÜR EIN ELEND":PRINT LEFT$(W$,3) <RETURN>
G$ = "ABSTAMMUNG":PRINT MID$(G$,3,5) <RETURN>
X$ = "KEIN PLATZ FÜR TIERE":PRINT RIGHT$(X$,19):PRINT RIGHT$
(X$,4) + MID$(X$,16,4) <RETURN>
```

Ein weiterer Trick mit Teilstrings ist die Zuweisung eines Teils des Strings zu einem anderen String. Zum Beispiel:

```
10 PRINT "<CLR/HOME>"
20 BIG$ = "LANG LANG HER UND WEIT WEIT WEG"
30 LIT$ = MID$(BIG$,12,2)
40 AWY$ = RIGHT$(BIG$,4)
50 LGE$ = LEFT$(BIG$,4)
60 PRINT:PRINT:PRINT LGE$;LIT$ + "" + AWY$
70 REM VOR DEM LAUFENLASSEN BITTE VERSUCHEN, DEN NEUEN TEXT
HERAUSZUFINDEN
```

Probieren Sie folgendes Programm, um einen interessanten Effekt zu erzielen:

```
10 PRINT "<CLR/HOME>":FOR I =1 TO 10:PRINT:NEXT
20 INPUT "IHR NAME -->";NA$
30  FOR I = LEN (NA$) TO 1 STEP -1:PRINT MID$(NA$,I,1);:
NEXT I
40 FOR I = 1 TO 1500:NEXT I:REM ANHALTEN DER ANZEIGE (PAUSE)
45 REM **ZEILE 50 BENUTZT <CLR/HOME> OHNE <SHIFT>
```

```
46 REM **BEACHTEN SIE DIE WIRKUNGSWEISE DER VERTIKALEN
CURSORBEWEGUNG
47 REM **IN ZUSAMMENHANG MIT DER SCHLEIFE
50 PRINT "<HOME>":FOR V = 1 TO 11:PRINT:NEXT V
55 REM **DIE K-SCHLEIFE IN ZEILE 60 DIENT DER VERLANGSAMUNG
60 FOR I = 1 TO LEN(NA$):PRINT MID$(NA$,I,1);:FOR K = 1 TO
60:NEXT K:NEXT I
70 FOR VT = 1 TO 5:PRINT:NEXT VT:PRINT TAB(5);"WOLLEN SIE
WEITERMACHEN? (J/N) ";
80 GET AN$:IF AN$ = "" THEN 80
90 IF AN$ = "J" THEN 10
```

Sie haben sich vielleicht gewundert, wie Ihr Computer es schafft, Namen rückwärts auszugeben. Nun, jetzt wissen Sie es! (Wenn Ihr Name OTTO ist, haben Sie es gar nicht bemerkt - probieren Sie deshalb OTTOKAR.) Das Programm führte aber auch noch andere Dinge durch: als erstes demonstrierten wir, wie Teilstrings und Schleifen zusammen benutzt werden können, um die Ausgabe zu formatieren. Zweitens zeigten wir, wie wir die Ausgabe verlangsamen können, um entweder einen interessanten Effekt zu erzielen oder dem Anwender Zeit zur Beobachtung zu geben.

Da wir gerade beim Thema Geschwindigkeit sind, lernen wir gleich, wie wir die Uhr des COMMODORE-64 benutzen können. Erinnern Sie sich an den Hinweis, daß TI$ eine "reservierte Variable" ist. Jetzt sehen wir warum. Probieren Sie folgendes im Kommandomodus:

```
TI$ = "101030" <RETURN>
```

Warten Sie ein paar Sekunden und geben Sie ein:

```
PRINT TI$ <RETURN>
```

Der eingegebene Wert von TI$ hat sich verändert! Haben Sie nur ein paar Sekunden gewartet, wurde aus 101030 101050 oder irgend etwas in diesem Bereich. Um zu sehen, was passiert, lassen Sie uns die Zahl in Stunden, Minuten und Sekunden teilen.

```
10 10 30 = 10 Uhr 10 Minuten 30 Sekunden.
```

Bei einer normalen Uhr würden wir sagen, es ist 10:10 und 30 Sekunden. Nun, TI$ macht das gleiche. Es zählt die Sekunden, dann die Minuten und zuletzt die Stunden. Um dies besser zu sehen, schreiben wir ein kleines Programm dafür.

```
10 PRINT"<CLR/HOME>":PRINT "COMMODORE-64 UHR"
20  FOR I = 1 TO 4:PRINT:NEXT:PRINT "GEBEN SIE DIE ZEIT  EIN
(00 STD 00 MIN 00 SEC)"
30 INPUT TI$
40 PRINT "<CLR/HOME>"
50  PRINT  "<HOME>":FOR  I  =  1  TO  10:PRINT:NEXT:PRINT
"COMMODORE ZEIT --> ";TI$:GOTO 50
```

Wenn Sie dieses Programm laufen lassen, vergewissern sie sich, daß Sie alle sechs Einheiten für Stunde, Minute und Sekunde eingegeben haben. Zum Beispiel bei 8:14 geben Sie 081400 ein und nicht nur 814.

Außer für die Ausgabe der Uhrzeit auf dem Bildschirm können Sie TI$ auch als Zeitauslöser im Programm verwenden. Sie weisen der Variablen TI$ zuerst einen Wert zu und fragen diesen im Programm ab. Sie können so die Zeit bis zur Eingabe einer Antwort messen. Das folgende kleine Mathematikprogramm enthält auch ein Beispiel zur Zeitmessung:

```
10 PRINT "<CLR/HOME>"
15 FOR I = 1 TO 5:PRINT:NEXT: TI$ = "000000"
20 INPUT "BITTE EINE ZAHL EINGEBEN ->";A
30 INPUT "NOCH EINE ZAHL           ->";B
40 PRINT:PRINT "WAS IST ";A;" + ";B;" "
50 INPUT C
60 IF A + B <> C THEN 200
70 IF TI$ > "000010" THEN GOTO 100
80 PRINT:PRINT "RICHTIG!!!!"
85 FOR X=1 TO 1000:NEXT:GOTO 10
100 PRINT "<CLR/HOME>":PRINT:PRINT "ZEIT ABGELAUFEN!"
110 FOR TM = 1 TO 1000:NEXT TM:GOTO 10
200 PRINT "DAS IST NICHT GANZ RICHTIG"
210 INPUT "BITTE <RETURN> DRÜCKEN";CR
220 GOTO 10
```

Gehen Sie das Programm aufmerksam durch. Beachten Sie, wie in Zeile 70 die Zeit geprüft wird und wie diese auf "000000" zurückgesetzt wird, wenn das Programm neu durchlaufen wird.

Umwandeln von Text in Zahlen und wieder zurück

Wir lernen jetzt Strings in Zahlen, und Zahlen in Strings zu verwandeln. Vielleicht denken Sie wie ich, als ich zum ersten Mal von diesen Kommandos hörte: das ist doch ziemlich überflüssig. Das gilt aber nur, wenn Sie Strings als Stringvariablen und Zahlen als numerische Variablen benutzen möchten. Wenn Sie jedoch einmal ihren Wert erkannt haben, werden Sie sich wundern, wie Sie jemals ohne diese Kommandos ausgekommen sind. Probieren Sie folgendes Programm:

```
10 PRINT "<CLR/HOME>"
20 FOR I = 1 TO 5:READ NA$(I):NEXT I
30 FOR I = 1 TO 5
40 X(I) = VAL(RIGHT$(NA$(I),1))
50 NEXT I
60 FOR I = 1 TO 5
65 PRINT "AUFGELAUFENE ZAHLUNGEN = DM";X(I)*(1.5*7)
67 NEXT I
70 DATA SMITH 7, JONES 8, MCKNAP 6, JOHNSON 2, KELLY 3
```

Unter Verwendung von DATA-Zeilen, die zuerst im Stringformat gespeichert waren, gelang es uns, Teile dieser Tabellenelemente in numerische Tabellenelemente umzuwandeln. Nachdem wir diese Konvertierung durchgeführt haben, konnten wir unsere mathematischen Operationen in Zeile 60 dazu benutzen, die aufgelaufene Summe zu errechnen, die jemand bei einer Auszahlung von 7 DM pro Stunde eineinhalbfach bekommt. Nun, das ist gut, sehr interssant, aber wir haben nun keine Liste, wer wieviel bekam und wie hoch die Gesamtsumme war. Warum probieren Sie das nicht allein? Ändern Sie das Programm so ab, daß der Name jedes Arbeitnehmers erscheint, mit der ausbezahlten Summe dahinter und der Gesamtsumme aller Beträge. (Tip: Sie suchen den Teilstring LEFT$(NA$(I),LEN (NA$(I))-2) da Sie die Zahl und die Leerstelle hinter jedem Namen brauchen.)

Es ist immer sehr hilfreich, mit den neuen Kommandos sofort einige Übungen durchzuführen, um sicherer zu werden. Probieren Sie:

```
A$ = "123":PRINT VAL(A$) + 11 <RETURN>
Q$ = "99.5":PRINT VAL(Q$)*7 <RETURN>
VK$ = "44.95":PRINT "VERKAUFT ZUM HALBEN PREIS   DM";
VAL(VK$)/2 <RETURN>
DO$ = "$103.88):DN$ = "$1834":PRINT VAL(RIGHT$(DO$,6))
+ VAL(RIGHT$(DN$,5)) <RETURN>
```

Hinweis: Wenn Sie das kleine Beispiel auf Band oder Diskette sichern möchten, schreiben Sie nur die Zeilennummern vorne dazu und speichern Sie es dann ab.

Von Zahlen zu Strings

Also, machen wir das ganze jetzt anders herum. Sie haben gesehen, warum wir Strings zu Zahlen machen; wozu aber Zahlen in Strings umwandeln? Für die Konvertierung benutzen wir das STR$ Kommando. Sehen Sie sich als Beispiel das folgende Programm an:

```
10 PRINT "<CLR/HOME>"
20 PRINT "GEBEN SIE EINE ZAHL MIT 5 STELLEN "
30 INPUT "HINTER DEM DEZIMALPUNKT EIN: ";A
40 A$ = STR$(A)
50 PRINT:PRINT LEFT$(A$,4)
```

Wie Sie sehen, haben Sie damit die Zahl auf drei Zeichen, inclusive Dezimalpunkt, gekürzt. Ändern Sie in Zeile 40 LEFT$ in RIGHT$, und Sie erhalten die 4 (nicht 3) äußerst rechten Zeichen des Strings. In Wirklichkeit gibt Ihnen LEFT$ ebenfalls 4 Zeichen, aber das Zeichen ganz links ist für das positive oder negative Vorzeichen einer Zahl reserviert. Üben wir noch einige Beispiele im Kommandomodus, um eine bessere Vorstellung zu bekommen, und lassen Sie uns anschließend noch etwas Praktischeres machen.

```
A = 5.00:A$ = STR$(A):PRINT A$ <RETURN>
V = 2345:V$ = STR$(V):PRINT V$ <RETURN>
GELD = 22.36:GELD$ = STRING$(GELD):PRINT LEFT$(GELD$,2)
<RETURN>
```

Erinnern Sie sich an diese Kommandos, wenn Sie mit Dezimalstellen arbeiten. Sie werden sich als sehr nützlich erweisen.

Verbinden von Strings: Verkettung

Sie haben gesehen, wie Sie Teile eines Strings nehmen und auf dem Bildschirm ausgeben können. Nun werden wir Strings miteinander verbinden. Diese Funktion heißt VERKETTUNG. Sie wird möglich durch Benutzung des "+" Zeichens in Verbindung mit dem String. Beispiel:

```
10 PRINT "<CLR/HOME>"
20 INPUT "IHR NACHNAME ->";NN$
30 INPUT "IHR VORNAME  ->";VN$
40 NA$ = VN$ + NN$
50 PRINT NA$
```

Ein bißchen verwirrend, was? Sie haben jedoch gesehen, wie NN$ und VN$ zu einem langen String zusammengefaßt wurden. Ändern Sie nun Zeile 40 folgendermaßen:

```
40 NA$ = VN$ + " " + NN$
```

Wenn Sie jetzt RUN eingeben, erscheint Ihr Name richtig auf dem Bildschirm. Wir haben nicht nur Stringvariablen verknüpft, sondern auch Strings hinzugefügt. Es ist zum Beispiel ganz in Ordnung, folgendes zu machen:

```
PRINT "EINS" + "EINS" <RETURN>
```

Sie können mit EINSEINS nicht viel anfangen, sehen aber gut das Prinzip der Verkettung.

Ein Problem bei der Formatierung von Zahlen mit dem COMMODORE-64 ist das Abtrennen von Nullen am Ende. Probieren Sie folgendes:

```
PRINT 19.80
PRINT 5.00
```

Bei einer Verarbeitung mit Mark und Pfennig kann Ihnen dies einige Kopfschmerzen bereiten und es sieht nicht gerade gut aus. Sehen wir also, wie wir dieses Problem mit unseren STR$- und VAL-Kommandos beheben können.

```
10 PRINT "<CLR/HOME>"
20 PRINT "BITTE NUR PFENNIGBETRÄGE EINGEBEN!":PRINT:PRINT
30 INPUT "WIEVIEL PF HABEN SIE AUSGEGEBEN?";PF
40 T = T + PF
50 T$ = STR$(T)
60  T$  = "000" + T$:REM DIES DIENT ZUR VERGEWISSERUNG  DASS
LEN(T$) LANG GENUG IST
70  IF MID$(T$,(LEN(T$)-1),1) = "." THEN T$ = T$ +  "0":GOTO
90
80 IF MID$(T$,(LEN(T$)-2),1) <> "." THEN T$ = T$ + ".00"
90 PRINT:PRINT:PRINT "SIE HABEN JETZT ";  RIGHT$(T$,LEN(T$)-
3); "DM AUSGEGEBEN"
100  PRINT "DRÜCKEN SIE EINE TASTE UM FORTZUFAHREN ODER  'E'
FÜR ENDE";
110 GET R$:IF R$ = "" THEN 110
120 IF R$ = "E" THEN END
130 GOTO 10
```

Dies sieht vielleicht etwas schwierig aus. Lassen Sie uns das Beispiel daher im einzelnen durchgehen:

1. Sie geben in Zeile 30 numerische Variablen ein und errechnen in Zeile 40 ihre Summe.

2. Die Summe, dargestellt durch die Variable T, wird dann in Zeile 50 in die Variable T$ umgewandelt.

3. In Zeile 60 "erweitern" Sie T$ durch 3 Nullen, um eine minimale Länge für Zeile 70 und 80 zu bekommen.

4. Zeile 70 errechnet das vorletzte Zeichen in T$. Wenn dieses Zeichen ein Dezimalpunkt (.) ist, wissen Sie, daß die zweite Dezimalstelle abgetrennt wurde. (z.B.

5.4, 19.5, usw.). Sie hängen also eine Null hinten an und springen zu Zeile 90.

5. Zeile 80 errechnet das dritte Zeichen von hinten und wenn dies kein Dezimalpunkt (.) ist, wissen Sie, daß alle Dezimalstellen abgetrennt wurden und es sich um einen reinen DM-Betrag handelt. Sie hängen also den Dezimalpunkt und zwei Nullen an (.00).

6. In Zeile 90 geben Sie schließlich die Ergebnisse aus, nachdem Sie die Erweiterung von Zeile 60 (000) unter Verwendung von RIGHT$ wieder abgetrennt haben. Die Formulierung 'LEN(T$)-3' errechnet die Länge von T$ und drei Stellen - die nicht gewünschten drei Nullen.

Das Ganze erscheint vielleicht als eine etwas komplizierte Methode, um unsere Nullen in die Dezimalstellen zurückzubekommen, aber in Wirklichkeit erstreckt sich der ganze Prozeß nur über fünf Zeilen (50 bis 90). Speichern Sie das Programm ab. Wenn Sie diese Anwendung in einer Ausgabe brauchen, fügen Sie nur diese Zeilen ein! (Vorsicht, die Subtraktion läuft nicht glatt ab, wenn Sie unter DM 1 kommen!).

5.3 Organisieren der Eingabedaten

Da Sie jetzt zahlreiche Kommandos fest im Griff haben, können Sie ernsthaft an die Organisation Ihrer Programme denken. Erstens müssen Sie Ihre Eingaberoutine in einer Weise arrangieren, daß Sie selbst und auch andere den Überblick nicht verlieren. Dies schließt das Anlegen von Blöcken im Programm und die Überlegung mit ein, welche Variablen und Tabellen Sie benutzen möchten. Ebenso möchten Sie sich bei der Dateneingabe vergewissern, daß Sie den richtigen Datentyp eingeben. Sie prüfen deshalb jede Eingabe anhand der Parameter daraufhin, daß die gewünschte Länge oder der Wert nicht überschritten werden. Sehen wir uns ein Beispiel an, wie die Strings alle in einer Länge einzugeben (nicht länger und nicht kürzer) sind. Wir haben bereits besprochen, wie wir die Länge auf ein Minimum festlegen. Der letzgenannte Prozeß wird auch als "padding" bezeichnet.

```
10 PRINT "<CLR/HOME>"
20 FOR I = 1 TO 8:PRINT:NEXT I
25 INPUT "IHRE FIRMA ->";CM$
30 IF LEN(CM$) < 10 THEN 70
40 IF LEN(CM$) > 10 THEN PRINT "BITTE NICHT MEHR ALS 10 ZEICH
EN EINGEBEN ":REM PRÜFUNG AUF ZU LANGE EINGABE
42 REM DRÜCKEN SIE <CTRL> UND <9> GLEICHZEITIG IN ZEILE 45
45 PRINT:PRINT "<CTRL><9> DRÜCKEN SIE BITTE EINE TASTE ->;
50 GET A$:IF A$ = "" THEN 50
60 GOTO 10
70  IF  LEN(CM$)  <  10 THEN CM$ =  CM$  +  "X":GOTO  70:REM
ANFÜGEN
80 PRINT "<CLR/HOME>"
90 FOR I = 1 TO 8:PRINT:NEXT
95 PRINT "DER COMPUTER HAT ENTSCHIEDEN, DASS "
99 PRINT CM$; " IHNEN EINE GEHALTSERHÖHUNG GEBEN SOLLTE!"
```

Wenn IHRE FIRMA <CM$> nun kleiner als 10 Zeichen ist, werden Sie einige "X" am Ende des Namens sehen. Wir haben dies gemacht, damit Sie sehen, wie die Verknüpfung arbeitet. Tauschen Sie nun in Zeile 70 das "X" durch ein Leerzeichen "" aus und sehen Sie, was passiert. Wenn Sie nun das Programm zum zweitenmal laufen lassen, merken Sie, daß einem Firmennamen, der kürzer als 10 Zeichen ist, mehrere Leerzeichen angehängt sind. Um diese Leerzeichen zu entfernen, geben Sie ein:

```
75 IF MID$(CM$,LEN(CM$),1) = " " THEN CM$ = LEFT$(CM$,(LEN
(CM$)-1):GOTO 75
```

5.4 Organisieren der Datenverarbeitung

Der nächste größere Schritt nach der Formatierung der Eingabe ist die Durchführung von Berechnungen mit den Daten. Sie werden hauptsächlich mit zwei Arten von Daten arbeiten:

1. NUMERISCH - Verarbeitung von numerischen Daten mit mathematischen Operationen

2. STRINGS - Verarbeitung von Zeichenketten mit Verknüpfungs- und Teilkettenkommandos.

Die meisten Stringverarbeitungen dienen der Formatierung der Ein- und Ausgabe. Wir werden daher die Formatierung numerischer Daten bevorzugt behandeln. Wir gehen hier ein einfaches Beispiel durch, in dem drei Verarbeitungen durchgeführt werden: (1) Additionen, (2) Subtraktionen und (3) fortlaufender Saldo. Es handelt sich um unser Scheckbuchprogramm aus früheren Kapiteln.

```
10 PRINT "<CLR/HOME>"
20 REM ### BEGINN DES EINGABE- UND ÜBERSCHRIFTENBLOCKS
30 CB$ = " = COMPUTER - SCHECKBUCH = ":L = 20 - LEN(CB$)/2
32 PRINT TAB(L);"<CTRL><9>";CB$:REM = ÜBERSCHRIFT =
40 FOR I = 1 TO 4:PRINT:NEXT
42 INPUT "GEBEN SIE DEN GEGENWÄRTIGEN SALDO EIN - DM ";BA
50 PRINT:PRINT:PRINT "1. EINLAGEN EINGEBEN ":PRINT
52 PRINT "2. SCHECKS ABZIEHEN "
55 PRINT:PRINT "3. PROGRAMMENDE"
60 FOR I = 1 TO 7:PRINT:NEXT
62 PRINT "<CTRL><9> VERARBEITUNG AUSWÄHLEN ";:INPUT A
70 ON A GOTO 100,200,400
80 GOTO 60:REM PRÜFUNG
90 REM ENDE DES EINGABEBLOCKS
100 REM ### DATENVERARBEITUNGSROUTINE NR. 1 ###
110 PRINT "<CLR/HOME>"
112 FOR I = 1 TO 6:PRINT:NEXT:INPUT "EINLAGE DM";DP
120 BA = BA + DP:REM SALDO ERMITTELM
130 PRINT:PRINT "SIE HABEN NUN  DM";BA;" AUF DEM KONTO"
140 PRINT:INPUT "<CTRL><9>WEITERE EINLAGEN? (J/N) ";AN$
150 IF AN$ = "J" THEN 110
160 PRINT:INPUT "HABEN SIE SCHECKS ZUM VERBUCHEN? (J/N)";AN$
170 IF AN$ = "N" THEN GOTO 400
180 IF AN$ = "J" THEN GOTO 200
190 PRINT "<CLR/HOME>"
192 GOTO 160:REM PRÜFUNG UND ENDE ROUTINE 1
200 REM ### DATENVERARBEITUNGSROUTINE NR. 2 ###
210 PRINT "<CLR/HOME>":FOR I = 1 TO 6:PRINT:NEXT
212 INPUT "BITTE SCHECKBETRAG EINGEBEN DM";CK
220 BA = BA - CK:REM SALDO ERMITTELN
230 PRINT:PRINT "SIE HABEN NUN  DM";BA;" AUF IHREM KONTO"
240 PRINT:INPUT "WEITERE SCHECKS? (J/N) - 'E' ENDE ";AN$
250 IF AN$ = "J" THEN 210
260 IF AN$ = "E" THEN 400
270 PRINT:INPUT "IRGENDWELCHE EINLAGEN? (J/N) ";AD$
280 IF AD$ = "J" THEN 100
290 GOTO 240:REM PRÜFUNG UND ENDE ROUTINE 2
```

```
400 REM ### ENDEVERARBEITUNG ###
410 PRINT "<CLR/HOME>":FOR I = 1 TO 400:PRINT "DM ";:NEXT
420 PRINT "IHR NEUESTER KONTOSTAND BETRÄGT  DM";BA
```

Dieses Programm wurde entworfen, um an einem Beispiel zu zeigen, wie man die Verarbeitung der Daten in sinnvolle Blöcke teilt. Mit der Datenausgabe gibt es jedoch noch einige Probleme. Wir bekommen nämlich keine Nullen am Ende des Saldos! Dies ist ein Ausgabeproblem, das wir im folgenden Abschnitt besprechen werden. Bevor wir aber fortfahren, sehen Sie bitte, ob Sie verstanden haben, wie wir die Verarbeitung der Daten in Blöcke geteilt haben. Wir haben nur drei Variablen benutzt:

```
BA = SALDO
CK = SCHECK
DP = EINLAGE
```

Bei der Verrechnung eines Schecks haben wir CH von BA subtrahiert. Bei der Eingabe einer Einlage haben wir DP zu BA addiert. So konnten wir immer einen fortlaufenden Saldo führen, und am Ende der Verarbeitung stellte BA den endgültigen Saldo dar. Durch das Einhalten der Blöcke waren wir in der Lage, im Programm zu springen und es dennoch geradlinig zu durchlaufen.

5.5 Organisation der Ausgabe

Lassen Sie uns zum Programm zurückkehren und es so verändern, daß unser Saldo die gewohnten Nullen aufweist. Dies ist hauptsächlich ein Problem der Ausgabe, nachdem alle Berechnungen zu diesem Zeitpunkt bereits erfolgt sind. Wir müssen jedoch nicht bei jeder Ausgabe des fortlaufenden Saldos die entsprechenden Zeilen für die Konvertierung des Saldos in eine Stringvariable eingeben. Wir geben diese Zeilen in einen Block, den wir bei Bedarf durchlaufen. Wenn Sie sich das Programm COMPUTER-SCHECKBUCH ansehen, stellen Sie fest, daß Sie ab Zeile 300 noch einen Block einfügen können. Benutzen Sie folgenden Block, um die Ausgabe zu formatieren:

```
300 REM ### FORMATIEREN DER AUSGABE ###
310 BA = BA + .001:PLACE = 1:BA$ = STR$(BA)
312 IF BA < .001 THEN BA$ = "0.00":GOTO 340
```

```
320 IF MID$(BA$,PLACE,1) <> "." THEN PLACE = PLACE + 1:GOTO
320
330 BA$ = LEFT$(BA$,PLACE + 2)
340 RETURN
350 REM ENDE DES AUSGABEBLOCKS
```

Wir ändern jetzt nur ein paar Zeilen in unserem Programm. Jedesmal wenn der Saldo ausgegeben werden soll, springen wir zu der Unterroutine in Zeile 300 und mit RETURN zurück zu der Ausgabe von BA$. Die folgenden Zeilen unseres COMPUTER-SCHECKBUCHS sollten geändert und/oder hinzugefügt werden:

```
125 GOSUB 300
130 PRINT:PRINT
132 PRINT "SIE HABEN NUN ";BA$;"DM AUF IHREM KONTO"
225 GOSUB 300
230 PRINT "SIE HABEN NUN ";BA$;"DM AUF IHREM KONTO"
415 GOSUB 300
420 PRINT "SALDO DM ";BA$
```

Haben Sie alles richtig eingegeben, steht Ihnen jetzt ein handliches, kleines Programm zur Führung Ihres Kontos zu Verfügung. Zur Überprüfung, ob alles vollständig ist, folgt anschließend noch einmal das gesamte Programm mit allen Unterroutinen und Änderungen, die wir durchgeführt haben:

```
10 PRINT "<CLR/HOME>"
20 REM ### BEGINN DES EINGABE- UND ÜBERSCHRIFTENBLOCKS
30 CB$ = " = COMPUTER - SCHECKBUCH = ":L = 20 - LEN(CB$)/2
32 PRINT TAB(L);"<CTRL><9>";CB$;REM = ÜBERSCHRIFT =
40 FOR I = 1 TO 4:PRINT:NEXT
42 INPUT "GEBEN SIE DEN GEGENWÄRTIGEN SALDO EIN - DM ";BA
50 PRINT:PRINT:PRINT "1. EINLAGEN EINGEBEN ":PRINT
52 PRINT "2. SCHECKS ABZIEHEN "
55 PRINT:PRINT "3. PROGRAMMENDE"
60 FOR I = 1 TO 7:PRINT:NEXT
62 PRINT "<CTRL><9> VERARBEITUNG AUSWÄHLEN ";:INPUT A
70 ON A GOTO 100,200,400
80 GOTO 60:REM PRÜFUNG
90 REM ENDE DES EINGABEBLOCKS
100 REM ### DATENVERARBEITUNGSROUTINE NR. 1 ###
110 PRINT "<CLR/HOME>"
112 FOR I = 1 TO 6:PRINT:NEXT:INPUT "EINLAGE DM";DP
120 BA = BA + DP:REM SALDO ERMITTELM
```

```
125 GOSUB 300
130 PRINT:PRINT
132 PRINT "SIE HABEN NUN ";BA$;"DM AUF IHREM KONTO"
140 PRINT:INPUT "<CTRL><9>WEITERE EINLAGEN? (J/N) ";AN$
150 IF AN$ = "J" THEN 110
160 PRINT:INPUT "HABEN SIE SCHECKS ZUM VERBUCHEN? (J/N)";AN$
170 IF AN$ = "N" THEN GOTO 400
180 IF AN$ = "J" THEN GOTO 200
190 PRINT "<CLR/HOME>"
192 GOTO 160:REM PRÜFUNG UND ENDE ROUTINE 1
200 REM ### DATENVERARBEITUNGSROUTINE NR. 2 ###
210 PRINT "<CLR/HOME>":FOR I = 1 TO 6:PRINT:NEXT
212 INPUT "BITTE SCHECKBETRAG EINGEBEN DM";CK
220 BA = BA - CK:REM SALDO ERMITTELN
225 GOSUB 300
230 PRINT "SIE HABEN NUN ";BA$;"DM AUF IHREM KONTO"
240 PRINT:PRINT "WEITERE SCHECKS? (J/N) - 'E' ENDE ";AN$
250 IF AN$ = "J" THEN 210
260 IF AN$ = "E" THEN 400
270 PRINT:INPUT "IRGENDWELCHE EINLAGEN? (J/N) ";AD$
280 IF AD$ = "J" THEN 100
290 GOTO 240:REM PRÜFUNG UND ENDE ROUTINE 2
300 REM ### FORMATIEREN DER AUSGABE ###
310 BA = BA + .001:PLACE = 1:BA$ = STR$(BA)
312 IF BA < .001 THEN BA$ = "0.00":GOTO 340
320 IF MID$(BA$,PLACE,1) <> "." THEN PLACE = PLACE + 1:GOTO
    320
330 BA$ = LEFT$(BA$,PLACE + 2)
340 RETURN
350 REM ENDE DES AUSGABEBLOCKS
400 REM ### ENDEVERARBEITUNG ###
410 PRINT "<CLR/HOME>":FOR I = 1 TO 400:PRINT "DM ";:NEXT
415 GOSUB 300
420 PRINT "SALDO DM ";BA$
```

5.6 Rollen des Bildschirms

Ein Problem der Ausgabe entsteht bei langen Listen, die nach oben aus dem Bildschirm rollen. Die Ausgabe des folgenden Programms z.B. wird oben aus dem Bildschirm hinausgerollt.

```
10 PRINT "<CLR/HOME>"
20 FOR I = 1 TO 100:PRINT I:NEXT
```

Stellen Sie sich vor, daß Sie anstelle der Zahlen sortierte Namen oder eine andere Ausgabe haben, die Sie sehen möchten, bevor sie aus dem Monitor verschwindet. Je nachdem, welche Ausgabe gewünscht wird, gibt es unterschiedliche Methoden das Rollen des Bildschirms zu kontrollieren. Überlegen Sie folgendes:

```
10 PRINT "<CLR/HOME>"
20 FOR I = 1 TO 100
30 IF I = 20 THEN GOSUB 100
40 IF I = 40 THEN GOSUB 100
50 IF I = 60 THEN GOSUB 100
60 IF I = 80 THEN GOSUB 100
70 PRINT I:NEXT I
80 END
100 PRINT:PRINT
110 PRINT "<CTRL><9>WEITER BEI BETÄTIGUNG EINER TASTE";
120 GET A$:IF A$ = "" THEN 120
130 PRINT "<CLR/HOME>":RETURN
```

Erinnern Sie sich: Sie (und nicht Ihr Computer) führen die Kontrolle durch! Sie können die Ausgabe ganz nach Wunsch gestalten. Um den Bildschirm besser zu nutzen, können Sie die Ausgabe auf einer anderen Spalte festlegen, nachdem der vertikale Bildschirm gefüllt ist. Zum Beispiel:

```
10 PRINT "<CLR/HOME>"
20 FOR I = 1 TO 40
30 IF I > 20 THEN GOSUB 100
50 PRINT I:NEXT I
80 END
100 PRINT "<HOME>":FOR J = 1 TO (I-20):PRINT:NEXT J
110 PRINT TAB(10);
120 RETURN
```

Formatieren Sie Ihre Ausgabe so, daß der Bildschirm am besten ausgenutzt ist und Ihren Bedürfnissen entspricht. Spielen Sie damit und Sie bekommen so das Rollen des Bildschirms gut unter Kontrolle.

5.7 Zusammenfassung

Der Unterschied zwischen einer nützlichen und einer nutzlosen Anwendung auf Ihrem Computer liegt in der Formatierung Ihrer Programme. Je besser und klarer Ihr Programm organisiert ist, desto größer ist die Chance, einfach und effektiv zu programmieren. Formatierung ist mehr als eine Übung, um die Ein- und Ausgabe interessant zu machen. Es geht um die Kommunikation zwischen Ihnen und Ihrem COMMODORE-64! Übrigens: wenn Sie keine Vor- und Nachläufe zu Ihrer Verarbeitung erstellen, sind die besten Berechnungen nicht zu gebrauchen.

Genauso wichtig ist es, daß Sie dem Computer mitteilen können, was Sie möchten. Es ist also von wesentlicher Bedeutung, die Programme so zu schreiben, daß Sie und andere verstehen, wie es abläuft. Durch Verwendung von "Blöcken" ist es einfacher, ein Programm zu organisieren, und es ist später besser nachzuvollziehen, was in jedem Teil des Programms genau passiert. Es ist selbstverständlich auch möglich, Programme fortlaufend in einer aufsteigenden Zeilennumerierung zu schreiben; Sie müssen dann aber sehr wahrscheinlich einfache und/oder komplexere Operationen wiederholt programmieren, die eigentlich besser in einer Unterroutine gehandhabt werden könnten. Es ist auch bedeutend schwerer, Fehler zu lokalisieren und die nötigen Änderungen vorzunehmen. Mit anderen Worten: durch die Benutzung einer strukturierten Vorgehensweise beim Programmieren wird die Erstellung des Programms einfacher, nicht komplizierter.

Schließlich sollten Sie auch verstehen, warum es TABs und Kommandos für Teilstrings gibt. Es sind nützliche Elemente, mit denen die verschiedenen Programmteile so organisiert werden können, daß Sie eine vollständige Kontrolle über die Ausgabe Ihres Computers haben. Was zuerst nur als nettes, einfaches Kommando im COMMODORE-64 stand, entwickelt sich über eine nützliche Anwendung zu einem hervorragenden Werkzeug. Wenn wir deshalb in Zukunft tiefer in die Möglichkeiten des Computers einsteigen, betrachten Sie die Vielalt der Kommandos als Mechanismus für eine effizientere und letztlich einfachere Kontrolle und nicht als fachspezifischen "Berufsjargon" von Computergenies. Übrigens: da Sie jetzt schon so weit sind, sollten Sie sich im klaren darüber sein, daß das, was Sie jetzt bereits wissen, am Anfang Ihrer Arbeit mit dem Computer wie ein "Wunder" ausgesehen hat.

6

Fortgeschrittene Anwendungen

Einführung

Die Themen dieses Kapitels betreffen den "Code" und enthalten die Art von Kommandos, vor denen man leicht zurückschreckt. Die meisten Funktionen können mit den uns bereits bekannten Kommandos durchgeführt werden, einige jedoch nicht. Wieder andere können, wie wir noch sehen werden, besser mit diesen neuen Kommandos formuliert werden. Wie so vieles in diesem Buch, was zuerst "unmöglich" zu lösen schien, ist wirklich sehr einfach, wenn Sie erst einmal wissen, wie es geht. Sehr wichtig ist es, mit den Kommandos zu spielen. Dadurch lernen Sie sehr schnell deren Anwendung.

Das erste, was wir hier lernen, ist der ASCII (ausgesprochen ÄS-KY), das für "AMERIKAN STANDARD CODE for INFORMATION INTERCHANGE" steht. Im Prinzip ist dies eine Reihe von Zahlen, die standardisiert wurden und die verschiedene Zeichen darstellen. Im COMMODORE-64-BASIC wird das CHR$ (Character string)-Kommando dazu benutzt, Zeichen direkt über ihren ASCII-Wert zu ermitteln und auszugeben. Wir werden sehen, daß das CHR$-Kommando zur Ausgabe von speziellen Zeichen sehr nützlich ist.

Die nächsten Kommandos ermöglichen den direkten Zugang zu Bereichen im Hauptspeicher des Computers. Das erste, POKE, hinterlegt Werte im Speicher. Das zweite, PEEK, sucht Speicheradressen und gibt die dort stehenden Werte zurück. Wir werden verschiedene Anwendungen zu diesen zwei Kommandos durchgehen. Diese Kommandos dienen hauptsächlich dazu, bestimmte Arten von Grafik und Tönen zu erzeugen.

6.1 Der ASCII-Code und die CHR$-Funktion

Wir benutzen in unseren Programmen Control Characters, wie z.B. CTRL-9. Es wird folgendermaßen am Bildschirm dargestellt:

```
PRINT "<INVERSES R>": REM CTRL-9
```

Dies bedeutet, daß wir CTRL-9 in Anführungszeichen eingeben, aber ein inverses R (dunkel auf hellem Hintergrund) erscheint. Leider können wir CTRL-9 nicht sehen, wenn wir unser Programm auf dem Drucker oder Bildschirm auflisten. Wir benutzen deshalb ein REM

(Bemerkung), um uns daran zu erinnern, daß ein inverses "R" in Wirklichkeit CTRL-9 ist. Ein anderer Weg, ein beliebiges Zeichen, (einschließlich Control-Zeichen), auszugeben, ist die Verwendung der CHR$-Funktion und des ASCII-Codes. In ANHANG A finden Sie eine komplete Liste aller ASCII-Zeichen. Immer wenn Sie ein Zeichen ausgeben möchten, nehmen Sie CHR$ und den dezimalen Wert des gewünschten Zeichens. Geben Sie zum Beispiel folgendes ein:

```
PRINT CHR$(65) <RETURN>
```

Sie erhalten damit ein "A". Das ist zwar einfach, aber nicht sehr interessant. Probieren Sie deshalb das folgende kleine Programm aus. Ich wette, Sie können es nicht ohne Verwendung der CHR$-Funktion schreiben.

```
10 PRINT CHR$(147):REM ASCII-CODE FÜR <CLR/HOME>
20 QU$ = CHR$(34):REM ASCII-CODE FÜR ANFÜHRUNGSZEICHEN "
30 FOR I = 1 TO 20:PRINT:NEXT
32 PRINT CHR$(18);"BITTE EINE BELIEBIGE TASTE DRÜCKEN ODER";
40 PRINT QU$;" E ";QU$;" ZUM BEENDEN"
50 GET AN$:IF AN$ = "" THEN 50
60 IF AN$ = "E" THEN END
70 GOTO 10
```

Geben Sie RUN ein und achten Sie genau auf die Ausgabe. Sie sehen die Anführungszeichen beim Q. Wenn wir versuchen, ein Anführungszeichen auszugeben, interpretiert der Computer dies als einen Ausgabestring. Da aber QU$ als CHR$(34) definiert wurde, können wir Anführungszeichen verwenden, ohne den Ausdruck durcheinander zu bringen (Probieren Sie nur zum Spaß mal aus, ob Sie die Ausgabe ohne die CHR$-Funktion erreichen).

Beachten Sie auch den Anfang des Programms. Anstatt des üblichen CLR/HOME haben wir CHR$(147) verwendet. Wir mußten dazu nicht die Anführungszeichen (") wie bei CLR/HOME benutzen. Ebenso haben wir CHR$(18) genommen statt CRTL-9, um den Inversmodus einzuschalten. Um zu sehen, welche unterschiedlichen Zeichen Sie verwenden können, geben Sie folgendes Programm ein:

```
10 PRINT CHR$(147)
20 FOR I = 32 TO 127:PRINT CHR$(I);:NEXT
30 FOR I = 158 TO 191:PRINT CHR$(I);:NEXT
```

Na also! Hier haben Sie alle Symbole. Bevor wir aber weitermachen, lassen Sie uns noch ein paar andere Symbole anschauen, die wir nur durch Drücken von zwei Tasten sichtbar machen können.

Halten Sie die COMMODORE-Taste (in der linken unteren Ecke der Tastatur) gedrückt und tippen Sie die SHIFT-Taste: die erste Reihe der Buchstaben wird in Kleinbuchstaben dargestellt und die Symbole, beginnend mit dem "Pik", verändern sich in Großbuchstaben. Je nachdem, ob die Kleinbuchstaben "ein"- oder "aus"-geschaltet sind, werden unterschiedliche Symbole ausgegeben. Tippen Sie nun folgende Zeilen ein:

```
10 PRINT CHR$(147)
20 FOR I = 0 TO 31
30 PRINT CHR$(I);:NEXT
```

Im Moment scheint nicht viel passiert zu sein - solange sich der Bereich der ASCII-Zeichen zwischen 0 und 31 befindet. In Wirklichkeit hat sich Ihr System "aufgehängt" und Sie müssen die RUN/STOP- und die RESTORE-Taste zusammen drücken, um neu zu starten.

Um in den Genuß der verstärkten Leistung Ihres Computers zu kommen, testen Sie folgende kleine Programme:

1. Programm

```
10 PRINT CHR$(147)
20 LB$ = CHR$(54):RB$ = CHR$(52)
30 CO$ = "COMMODORE" + CHR$(45) + LB$ + RB$
40 L = 20 - LEN(CO$)/2:PRINT SPC(L);CO$
50 FOR I = 1 TO 20:PRINT CHR$(32):NEXT
```

2. Programm

```
10 PRINT CHR$(147)
20 PRINT CHR$(18);CHR$(28);:FOR I = 1 TO 35:PRINT CHR$(32);
22 NEXT
30 PRINT CHR$(5):REM VERSUCHEN SIE BITTE VOR DER EINGABE
DIESES PROGRAMMS, DIE AUFGABE SO ZU LÖSEN
```

Im letzten Programm haben Sie einen Hinweis zum Gebrauch der CHR$-Funktion in Verbindung mit der Erzeugung einer Grafik erhalten. Der rote Balken wurde durch CHR$(32) erzeugt, das Leerzeichen dahinter durch CHR$(18) (entspricht <CTRL-9>) und CHR$(28) (entspricht <CTRL-3>). Im nächsten Grafikkapitel benutzen wir CHR$-Funktionen, um richtige Bilder, Tabellen und Grafiken zu erzeugen. (Übrigens, um wieder den normalen Modus zu erreichen, benutzen Sie die RUN/STOP- mit der RESTORE-Taste.)

Das folgende kleine Programm stellt eine Möglichkeit dar, alle Werte der CHR$-Zeichen am Bildschirm auszugeben. Sichern Sie es auf Diskette oder Band und Sie haben ein gutes Nachschlagewerk für alle Werte und die entsprechenden Zeichen und Symbole der ASCII-Tabelle.

CHR$-LISTE

```
10 PRINT CHR$(147)
20 GOSUB 300
30 FOR I = 33 TO 99
40 IF I = 34 THEN GOTO 400
50 PRINT I;". =";CHR$(I),
60 NEXT
70 PRINT:PRINT "ZUM FORTFAHREN IRGENDEINE TASTE BETÄTIGEN";
80 GET A$:IF A$ = "" THEN 80
90 PRINT CHR$(147)
100 GOSUB 300
110 FOR I = 100 TO 127:PRINT I;". =";CHR$(I),:NEXT
120 FOR I = 161 TO 191:PRINT I;". =";CHR$(I),:NEXT
130 PRINT:PRINT:END
300 FOR I = 1 TO 4:PRINT "CHR$/S",:NEXT
310 RETURN
400 PRINT I;". =";"'",:REM DAS AUSZUGEBENDE HOCHKOMMA MUSS
ZWISCHEN ZWEI EXTRA-HOCHKOMMAS EINGESCHLOSSEN SEIN
410 GOTO 60
```

Das Programm "CHR$-LISTE" kann als ein Nachschlagewerk benutzt werden, um sich die CHR$-Werte verschiedener Symbole anzusehen. Sie haben bemerkt, daß das Programm zu einer Unterroutine in Zeile 400 verzweigt, wenn I = 34 ist. Der Grund dafür ist das Anführungszeichen - CHR$(34); der Rest der Ausgabe würde nur Klammern invers darstellen. Um dies zu vermeiden, haben wir ein "fal-

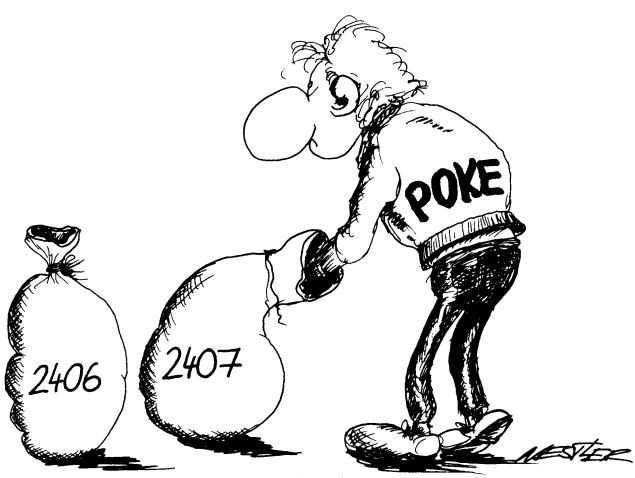
sches Anführungszeichen" unter Verwendung von zwei Apostrophen (SHIFT-7) benutzt. Es entsteht dadurch zwar eine Lücke zwischen 34 und 35, aber das sieht besser aus als die inversen Klammern. Ferner haben wir CHR$-Werte übersprungen, die den Bildschirm sperren, diesen löschen, die Farbdarstellung wechseln oder die Ausgabe durcheinander werfen. Sehen Sie zu, ob Sie ein Programm schreiben können, das nützliche CHR$-Werte (wie z.B. CTRL-9 und Farbattribute) enthält; zerstören Sie aber dabei nicht die Ausgabe.

6.2 POKE und PEEK

Ein Blick in den Speicher Ihres Commodore-64

Am Anfang werden Sie PEEK und POKE nicht häufig anwenden. Wenn Sie aber erst einmal die volle Leistungsfähigkeit Ihres Computers entdeckt haben, werden Sie diese Kommandos mehr und mehr in Ihre Programme einbeziehen. Grundsätzlich speichert das POKE-Kommando einen Wert am angegebenen Speicherplatz und ein PEEK-Kommando liest den Inhalt eines Bytes an der angegebenen Adresse. Versuchen Sie folgendes:

```
POKE 2048,255:PRINT PEEK(2048) <RETURN>
```



Sie sollten jetzt "255" sehen, da das POKE-Kommando diesen Wert auf den Speicherplatz 2048 brachte und PRINT PEEK(2048) den Inhalt dieser Adresse wieder ausgegeben hat. Das ist relativ einfach, aber es gibt noch mehr Möglichkeiten als das Speichern von Zahlen.

An manchen Stellen finden wir nur die abgespeicherte Zahl, wie in unserem Beispiel oben. An anderen Stellen jedoch löst POKE, je nach übertragenem Wert, genau definierte Ereignisse aus. Im folgenden wollen wir uns einige bedeutendere Speicherplätze unseres COMMODORE-64 für das PEEKen und POKEn ansehen. (Wir wollen jedoch zu komplexe Elemente der PEEKs und POKEs ausklammern).

Benutzung von zwei Zahlensystemen

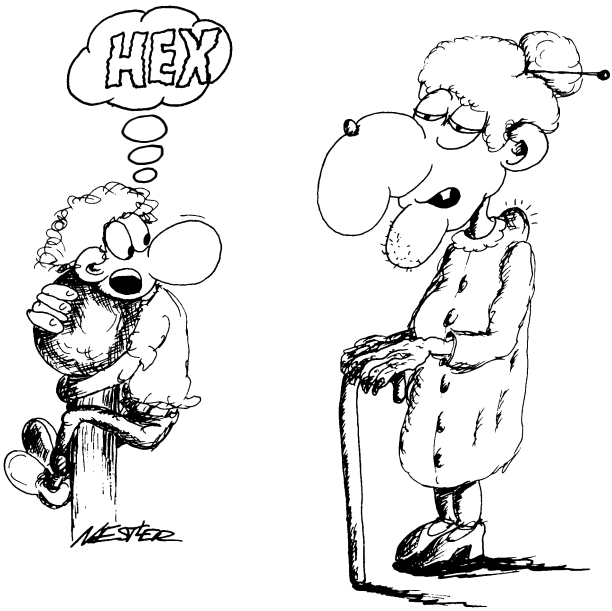
Wenn wir PEEK und POKE benutzen, nehmen wir dezimale Zahlen, um Speicherplätze zu verändern. Die meisten der speziellen Speicheradressen, die im COMMODORE-64-HANDBUCH beschrieben sind, werden HEXADEZIMAL (kurz HEX genannt) dargestellt. Da wir normalerweise immer im Dezimalsystem rechnen, erscheint uns dieses als naturgegeben. Das Dezimale System hat als Basis den Wert 10, es gibt jedoch auch Zahlensysteme mit ganz anderen Basen, etwa 8 oder 2. Die "Basis 16", genannt HEXADEZIMAL, bietet einen einfacheren Weg zur Berechnung des Speicherplatzes eines Computers. Aus diesem Grund sehen Sie viele der Angaben in HEX. Hexadezimal wird genau so gezählt wie dezimal, allerdings in 16er-Gruppen. Es benutzt außer den Zahlen auch noch alphanumerische Zeichen. Sie erkennen eine Zahl sofort als HEX, da diese üblicherweise mit einem Dollarzeichen dargestellt wird (z.B. ist $45 nicht dasselbe wie die dezimale Zahl 45). Oft sehen Sie auch alphanumerische Zeichen mit Zahlen zusammen dargestellt (z.B. FC58, AAB, 12C). Im Anschluß sehen Sie eine Liste mit dezimalen Zahlen und der entsprechenden hexadezimalen Darstellung.

DEZIMAL	HEXADEZIMAL
0	$0
1	$1
2	$2
3	$3
4	$4
5	$5
6	$6
7	$7
8	$8
9	$9
10	$A
11	$B
12	$C
13	$D
14	$E
15	$F
16	$10

Wie Sie sehen, beginnen beim hexadezimalen System zwei-stellige Zahlen nicht bei 10, sondern erst bei 16 (mit $10). Im Anhang Q des COMMODORE-64-HANDBUCHS sind die Adressen sowohl hexadezimal, als auch dezimal vermerkt.

Ein fauler Trick!

Wenn Sie beginnen, POKE und PEEK an verschiedenen Speicherstellen Ihres COMMODORE-64 anzuwenden, wird nicht immer das eintreten, was Sie erwarten. Zwischen den dezimalen Adressen 2048 und 40959 sind Sie vor unliebsamen Überraschungen ziemlich sicher, da es sich hier um den Bereich handelt, der vom BASIC belegt wird. In anderen Bereichen jedoch, befinden sich spezielle Routinen, die auf in sie gePOKEte Werte direkt reagieren. Wenn Sie z.B. POKE 768,255:PRINT PEEK(768) eingeben, wird Ihre Maschine hängenbleiben und nicht einmal die Tasten <RUN/STOP> und <RESTORE> bringen sie wieder zum Laufen. Sie müssen Ihren Computer ausschalten, um ihn aus diesem Zustand herauszuholen. Wenn Sie dies in einer Ihrer Programme einbauen und es anschließend einem Freund geben, würde auch seine Maschine hängenbleiben und das wäre ein fauler Trick. Natürlich würden Sie so etwas nie machen. Oder doch?



Lassen Sie uns nun einen Blick auf einige Speicherplätze werfen, die wir mit POKE beeinflussen können. Wir beginnen dazu mit Ihrem Bildschirmspeicher.

6.3 POKEn im Bildschirmspeicher

Eine andere Verwendung von POKE ist die Einspeicherung von Zeichen im Bildschirmspeicher. Jedes Zeichen hat einen ihm fest zugeordneten Wert zwischen 0 und 255. Sie können sich Ihren Bildschirm als eine Adressenfolge in einer 40 x 25-Matrix vorstellen, die bei Adresse 1024 beginnt und bei 2023 endet. Dadurch haben Sie genau 1000 Stellen auf dem Monitor, in die Text plaziert werden kann. Die Adressen sind fortlaufend angeordnet und können mit FOR/NEXT-Schleifen auf einfache Art mit fortlaufenden Textzeilen gefüllt werden. Sie können auch durch Anwendung von POKEs an beliebiger Stelle des Bildschirms Text einspeichern. Damit Sie eine Vorstellung davon bekommen, was Sie dann sehen werden, lassen Sie das folgende kleine Programm laufen und versuchen Sie dann die POKEs.

```
10 PRINT CHR$(147):B = 55296:FOR I = B TO B + 999:POKE I,1:
NEXT

POKE 1190,1:POKE 1191,129 <RETURN>
FOR I = 1880 TO 1890:POKE I,81:NEXT <RETURN>
FOR I = 1240 TO (1240 + 255):POKE I, I - 1240:NEXT <RETURN>
```

Die erste Zeile zeigt verschiedene Werte für ein normales und ein inverses "A", die auf einander folgende Adressen gepokt werden. Die nächste Übung benutzt eine Adressenfolge von 1880 bis 1890 und stellt einen "weißen Ball" an jede Speicherstelle. Die dritte Übung benutzt die angrenzenden Speicherstellen, um eine Reihe von ASCII Zeichen zu hinterlegen.

Das folgende Programm soll Sie in das "Konzept der Distanz" einweisen. Im Prinzip ist die Distanz ein veränderlicher Wert, der zu einem festen Anfangswert addiert oder von diesem subtrahiert wird. In unserem Programm sind zwei verschiedene Distanzen zu beachten. Die erste ist "127" und legt die maximale Adresse für die Programmschleifen in Zeile 20 und 40 fest. Da wir 128 Zeichen (von 0 bis 127) mit POKE einbringen möchten, setzen wir unsere erste Distanz auf 127 und berechnen unsere Bildschirmadresse aus Distanz plus Anfangsadresse. Da wir mit 0 starten (I-1024), enden wir mit 127, unserer festgelegten Distanzangabe. Zweitens benutzen wir in Zeile 50 eine Distanz von 128, um die im ersten Ab-

schnitt erzeugten inversen Zeichen zu erhalten. Das funktioniert, da jedes Zeichen von 0 bis 127, das wir mit POKE eingespeichert haben, ein inverses Gegenstück hat, dessen Wert um genau 128 höher liegt. Deshalb müssen wir, wenn wir ein Zeichen invers darstellen möchten, nur einfach 128 zum Original-Poke-Wert addieren.

```
10 PRINT CHR$(147)
20 FOR I = 1024 TO (1024 + 127)
30 POKE I,(I - 1024):NEXT
32 FOR X = 55296 TO (55296 + 127)
34 POKE X,1:NEXT
40 FOR I = 1424 TO (1424 + 127)
50 L = I - 1424:POKE I,L + 128:NEXT
52 FOR X = 55696 TO (55696 + 127)
54 POKE X,1:NEXT
60 FOR I = 1 TO 15:PRINT:NEXT
```

Sie wundern sich vielleicht, warum wir die Zeile 60 im Programm haben. Nehmen Sie diese heraus und sehen Sie, was passiert. Wir brauchen die Zeile 60, damit der Cursor nicht den Zeilennummern des Programms folgt, sondern der zu pokenden Bildschirmstelle. Sie können beispielsweise ein Zeichen an die HOME-Position (links oben) des Bildschirms poken und ebenso ein Zeichen in der letzten Zeile des Bildschirms ausgeben, der Cursor bleibt immer am oberen Bildschirmrand. Probieren Sie es aus.

Um ganz leicht zu sehen, welche Zeichen durch die verschiedenen Werte produziert werden, poken wir sie in den Bildschirmspeicher. Das folgende Programm ermöglicht es Ihnen, einen Wert einzugeben und anschließend das entsprechende Zeichen am Bildschirm einzusehen. Von besonderem Interesse sind die Zeilen 50 und 60. Zeile 50 gibt eine Nachricht aus und endet statt mit einem Semikolon mit einer Leerstelle. Geben Sie jedoch RUN ein, wird das Zeichen gleich anschließend an den String aus Zeile 50 ausgegeben, weil wir die Ausgabe auf dem Bildschirm direkt hinter den String gepokt haben. Daher ist es nicht von Bedeutung, ob sich am Ende der Zeile 50 ein Semikolon, ein Komma oder ein Leerzeichen befindet. Die Ausgabe ist immer an der gleichen Stelle. Sehen Sie selbst:

```
10 PRINT CHR$(147);CHR$(5)
20 PRINT:PRINT:INPUT "BITTE EINE ZAHL ZWISCHEN 0 UND 255
EINGEBEN ->";X
30 IF X > 255 THEN 20
40 PRINT CHR$(19):FOR I = 1 TO 11:PRINT:NEXT
```

```
50 PRINT "DAS ZEICHEN FÜR ";X;" IST "
60 POKE (1504 + 25),X:POKE (55776 + 25),1
70  PRINT:PRINT  CHR$(18);"WEITER MIT BELIEBIGER  EINGABE  -
ENDE = 'E'";
80 GET HK$:IF HK$ = "" THEN 80
90 IF HK$ <> "E" THEN 10
```

Fertigen Sie Listen an!

Zusätzlich zu den Beschriftungen, die ich an meinem Computer angebracht habe, besitze ich noch eine Menge von Aufstellungen. Gut daran ist, daß man auf diesen Listen, nach Kategorien geordnet, alles beieinander hat. Sie sollten sich eine Liste anfertigen oder kaufen, die alle SYS-, POKE- und andere nützliche Speicherplätze und Adressen enthält. Auch in vielen Computerzeitschriften finden Sie solche Auflistungen. Sammeln Sie alle, sie sind sehr von Nutzen.

MINI SYS-LISTE

SYS 58692	entspricht CLR/HOME
SYS 58726	entspricht HOME
SYS 59903	löscht ganze Textzeilen
SYS 59062	Cursor positionieren
SYS 59626	Bildschirm eine Zeile hochrollen
SYS 59137	zurück zur aktuellen Zeile

Unterroutinen in Maschinensprache

Das SYS-Kommando ist ein sehr nützliches Werkzeug zum Beschleunigen Ihrer Programme. Ein SYS-Kommando startet eine Betriebssystemroutine im ROM des Computers oder im Hauptspeicher. Im COMMODORE-64-HANDBUCH finden Sie eine Liste der ROM-Routinen Ihres Computers. Ganz links sehen Sie die Startadressen der verschiedenen Assemblerroutinen, die mit SYS vom BASIC-Programm aus aufgerufen werden können. Ein Problem stellen die hexadezimalen Anga-

ben dar, da beim SYS-Kommando grundsätzlich nur die dezimalen Werte verwendet werden. Es sind deshalb Programme und Tabellen zur Konvertierung von hexadezimal in dezimal vorhanden. Für den Anfänger wäre es nämlich eher verwirrend als klärend, sich den Umsetzungsprozeß oder das Monitorlisting näher anzusehen. Stattdessen geben wir Ihnen eine Liste mit einigen nützlichen SYS-Kommandos. Wenn Sie weiter fortgeschritten sind, werden Sie leichter verstehen, wie diese Befehle mit den Betriebssystemroutinen zusammenarbeiten.

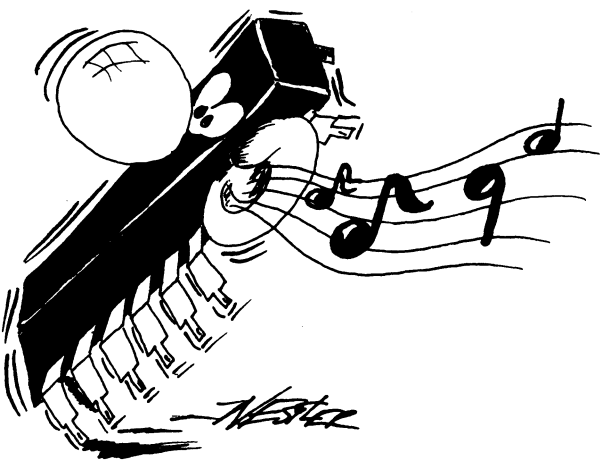
Probieren Sie die oben angegebenen SYS's in Ihren Programmen aus, um den Effekt zu sehen. Hier sind einige Beispiele dafür:

```
10 SYS 58692
20 FOR I = 1 TO 800:PRINT "X";:NEXT
30 REM BILDSCHIRM AUSFÜLLEN
40 SYS 58726:REM CURSOR HOME
50 FOR I = 1 TO 10:PRINT:NEXT

10 SYS 58692
20 FOR I = 1 TO 200:SYS 59062:NEXT
30 PRINT "WIE BIN ICH HIERHER GEKOMMEN?"

10 SYS 58962
20 FOR I = 1 TO 840:PRINT "&";:NEXT
30 FOR SCROLL = 1 TO 12
40 FOR PAUSE = 1 TO 400:NEXT PAUSE
50 SYS 59626:REM SCROLL
60 NEXT SCROLL
```

6.4 Eine ganze Menge Töne

Da Sie jetzt wissen, wie wir mit POKE Werte an spezielle Speicheradressen bringen können und sofort Ergebnisse erhalten, werfen Sie einen Blick auf die phantastischen musikalischen Möglichkeiten des COMMODORE-64. Man kann damit nicht nur Musik, sondern praktisch jeden gewünschten Ton erzeugen. Wir machen es uns jedoch einfach und zeigen Programme und Instruktionen, mit denen Ihnen lediglich der Einstieg in die Materie erleichtert werden soll. Dazu benutzen wir einen speziellen Chip im COMMODORE-64. Er wird als 6581-Chip oder SID-SOUND-SYNTHESIZER-CHIP bezeichnet.

Bis jetzt haben wir mit dem 6510-Microprozessor-Chip gearbeitet, dem "Herzen" Ihrer Maschine. Im Grunde poken wir Werte in den 6581 und bekommen Töne zurück zum Monitor. Bevor wir also weitermachen, drehen Sie den Ton des Monitors auf. (Haben Sie einen Monitor ohne Lautsprecher, können Sie diese Übungen nicht durchführen.) Speichern Sie die folgenden Übungen auf Diskette ab. So können Sie Noten oder Töne mit nur wenigen Zahlen eingeben.



Sehen wir uns zu Beginn die Speicherplätze an, die wir mit POKE ansprechen. Was geschieht dort?

1. LAUTSTÄRKE. Hier wird die Lautstärke auf das Maximum von 15 gesetzt. Dies ist für die Tonausgabe üblich. Als Variable benutzen wir VL für die Lautstärke.

2. ATTACK/DECAY und SUSTAIN/RELEASE. Diese zwei Angaben regeln, wie schnell ein Ton ansteigt und dann vom maximalen Lautstärken-Level abfällt (Attack/Decay) sowie die Dauer, die der Ton auf einem bestimmten Level gehalten wird, bevor dieser wieder verlassen wird (Sus-

tain/Release). Wir benutzen dazu die Variable AD für Attack/Decay und SR für Sustain/Release.

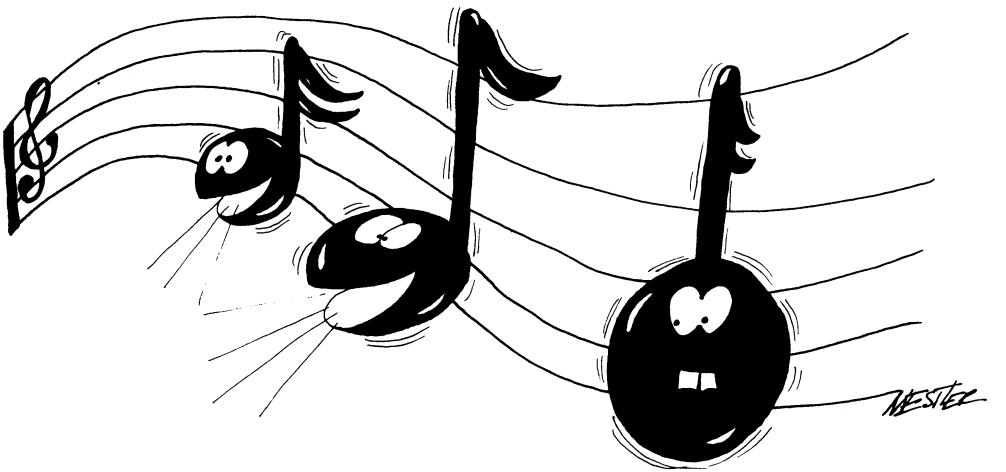
3. WELLENFORM. Die Wellenform entscheidet darüber, wie der Ton erzeugt wird. Wir benutzen nur vier Wellenformen: Triangel = 17, Sägezahn = 33, Puls = 65 und Rauschen = 129. Wir benutzen die Variable WF für Wellenform.

4. HOHE/NIEDRIGE FREQUENZEN. Diese Werte erzeugen einen spezifischen Ton. Jeder Ton erfordert einen hohen und einen niedrigen Wert. HF steht für die hohe Frequenz und LF für die niedrige.

Je nachdem, welche Werte in den angegebenen Variablen gepokt werden, werden verschiedene Töne ausgesendet. Wenn wir zum Beispiel VL=10 poken, ist der Wert der Lautstärke 10. Damit haben wir hohe/mittlere Lautstärken. Bevor wir die Variablen poken, müssen wir sie aber erst definieren. (Sichern Sie den ersten Teil des Programms unter dem Namen "TONVARIABLEN" auf Band oder Diskette, so daß Sie nicht alle Werte neu eingeben müssen, wenn Sie eine Ton- oder Musikroutine schreiben möchten.)

```
10 PRINT CHR$(147)
20 VL = 54296:REM LAUTSTÄRKE
30 LF = 54272:REM NIEDERFREQUENZ-TON
40 HF = LF + 1:REM HOCHFREQUENZ-TON
50 LW = LF + 2:REM LANGSAM SCHWINGENDER TON
60 HW = LF + 3:REM SCHNELL SCHWINGENDER TON
70 WF = LF + 4:REM SINUSTON
80 AD = LF + 5:REM ATTACK/DECAY
90 SR = LF + 6:REM SUSTAIN/RELEASE
```

Jetzt haben wir die Variablen definiert und können die Werte eingeben, die wir poken möchten. Um Ihnen eine Vorstellung davon zu geben, wie das funktioniert und welche Ergebnisse Sie mit den verschiedenen Werten erhalten, verwenden wir eine Reihe von INPUT-Anweisungen, damit Sie auf einfache Weise die Töne und Klänge testen können, die von den Werten erzeugt werden.



```
100 PRINT:PRINT
110 INPUT "LAUTSTÄRKE       (0-15)";V%
120 INPUT "ATTACK           (0-15)";A%
130 INPUT "DECAY            (0-15)";D%
140 INPUT "SUSTAIN          (0-15)";S%
150 INPUT "RELEASE          (0-15)";R%
160 INPUT "WELLENFORM       (0-3 )";W%:PRINT
165 IF W% <> 2 THEN PRINT:GOTO 180
170 INPUT "PULSBREITE       (0-20.47)";W1
180 INPUT "PITCH            (.01-655.35)";P1
190 INPUT "DAUER            (IN SEKUNDEN)";DU
```

Sie haben damit eine Routine, mit der Sie zu pokende Werte eingeben können. Was jetzt noch fehlt, ist die Ausgabe der Töne.

```
200 W = 16 * 2 ^ W %
205 W2% = W1 / 2.56:W3% = W1 * 100 - W2% * 256
210 P2% = P1 / 2.56:P3% = P1 * 100 - P2% * 256
215 POKE VL,V%
220 POKE AD,A% * 16 + D%
```

```
225 POKE SR,S% * 16 + R%
230 POKE WF,W + 1
235 POKE LW,W3%:POKE HW,W2% + 8
240 POKE LF,P3%:POKE HF,P2%
245 FOR T = 1 TO 800 * DU:NEXT
250 POKE WF,W
255 PRINT CHR$(19);"WEITER? (JA/NEIN)";
260 AN$= "JA":INPUT AN$
265 IF LEFT$(AN$,1) = "J" THEN 100
300 PRINT CHR$(19);
310 POKE LF,0:POKE HF,0
320 END
```

Vergewissern Sie sich, daß der Lautsprecher eingeschaltet ist und geben Sie RUN ein. Probieren Sie die verschiedensten Kombinationen aus, bis Sie den Ton bekommen, den Sie haben wollen.

6.5 Zusammenfassung

Dieses Kapitel hat uns in den Speicherbereich des COMMODORE-64 geführt. Wenn Sie auch noch kein Profi sind, haben Sie doch eine Ahnung davon bekommen, was ASCII-Werte bewirken und wissen jetzt ein wenig mehr über Adressen und Speicherbereiche. Hoffentlich haben Sie einige der gelernten Kommandos in Ihren Programmen ausprobiert. Je mehr unterschiedliche Kommandos Sie anwenden können, desto größer wird Ihr Verständnis für das Ganze.

Die CHR$-Funktion hat Sie mit den ASCII-Werten bekannt gemacht. Wir haben gelernt, daß wir mit dieser Funktion Zugriff zu verschiedenen Zeichen haben, die wir sonst in unseren Programmen nicht verwenden könnten.

Das POKE-Kommando hinterlegt einen Wert an einer dezimalen Adresse, das PEEK Kommando holt den Wert an dieser Adresse. Spezielle Bereiche im Speicher Ihres COMMODORE-64 haben spezielle Funktionen, wie etwa die Steuerung der Bildschirmausgabe. Fortgeschrittene Anwendungen von PEEK und POKE versetzen Sie in die Lage, Unterroutinen auf Maschinenebene schreiben zu können.

7

Benutzen der Grafik

Einführung

Eine der schönsten Funktionen des COMMODORE-64 ist die Möglichkeit, Grafik zu erzeugen. Es gibt zwei grundlegende Arten der Grafik: (1) die Bildschirmgrafik und (2) die Spritegrafik. Die Bildschirmgrafik ähnelt einem Text, man benutzt aber viel mehr Farben und Figuren anstatt Buchstaben und Zahlen. Man kann sowohl die vorhandenen Grafikzeichen als auch Texte gleichzeitig darstellen. Diese Möglichkeit ist vor allem gut, um eine Grafik zu beschriften, wenn man zum Beispiel Geschäftsgrafiken oder Figuren darstellen möchte. Drücken Sie die "Commodore-Taste" oder die SHIFT-Taste und eine weitere Taste gleichzeitig, so erhalten Sie die Computergrafik.

Die Spritegrafik unterscheidet sich vollständig von der Bildschirmgrafik. Ihre Anwendung ist auch sehr viel komplizierter. Pixelgrafik gibt Ihnen jedoch eine unglaubliche Flexibilität und Leistungsfähigkeit im Detail, speziell für die Anwendung in Spielprogrammen. Wenn Sie etwas erfahrener in der Anwendung der Pixelgrafik sind, sehen Sie die unbegrenzten Möglichkeiten für die Erzeugung von ansprechenden farbigen und bewegten Grafiken.

7.1 Bildschirmgrafik

Bildschirmgrafik ist sehr einfach in der Anwendung, da Sie die Zeichen direkt über die Tastatur eingeben. Sie geben eine einzelne Figur genau wie eine Zahl oder einen Buchstaben mit PRINT aus. Wenn Sie z.B. PRINT "<SHIFT-Z>" eingeben,

erhalten Sie das Karo-Zeichen. Um jedoch interessantere Grafik zu erzeugen, müssen Sie Kommandos im Programmodus eingeben. Eine Möglichkeit ist die Eingabe mehrerer PRINT-Anweisungen bei deren Ausführung nach und nach eine Zeichnung entsteht. Lassen Sie uns nun eine Spielkarte zeichnen. Wir machen es uns zu Beginn einfach und malen eine Pik-2. (Es wäre gut, das Programm auf Band oder Diskette abzuspeichern, wie auch die anderen Übungen dieses Kapitels. Sichern Sie diese unter verschiedenen Namen, auch wenn identische Ergebnisse erzeugt werden, da die Programmierung unterschiedlich ist.)

```
10 PRINT CHR$(147)
20 PRINT "<SHIFT-U> <7MAL SHIFT-D> <SHIFT-I>"
30 PRINT "<SHIFT-G> 2 <SHIFT-A> 5LEER <SHIFT-I>"
40 PRINT "<SHIFT-G> 7LEER <SHIFT-H>"
50 PRINT "<SHIFT-G> 3LEER <SHIFT-A> 3LEER <SHIFT-H>"
60 PRINT "<SHIFT-G> 7LEER <SHIFT-H>"
70 PRINT "<SHIFT-G> 3LEER <SHIFT-A> 3LEER <SHIFT-H>"
80 PRINT "<SHIFT-G> 7LEER <SHIFT-H>"
90 PRINT "<SHIFT-G> 5LEER 2 <SHIFT-A> <SHIFT-H>"
100 PRINT "<SHIFT-J> <7MAL SHIFT-F> <SHIFT-K>"
```

Haben Sie das Programm eingegeben, sollten Sie die vollständige Spielkarte bereits auf dem Bildschirm sehen, auch ohne die Eingabe von RUN. Geben Sie nun RUN ein, wird der Bildschirm gelöscht und die "Pik-2" erscheint links oben am Bildschirm. Auf die gleiche Art können Sie eine Vielzahl weiterer Zeichnungen mit Hilfe der verschiedenen Figuren Ihrer Tastatur erstellen. Erinnern Sie sich: um das rechte Zeichen auf der Taste zu bekommen, benutzen wir die SHIFT-Taste, zur Ausgabe des linken Tastenzeichens benutzen wir die COMMODORE-Taste.

Sehen wir uns noch einmal die "Pik-2" an, um zu klären, ob wir dieses Programm verbessern können. Beachten Sie als erstes, daß die Zeilen 40, 60 und 80 identisch sind, ebenso wie Zeilen 50 und 70. Anstatt die Zeilen wiederholt einzugeben, benutzen Sie jetzt das GOSUB-Kommando und behandeln Sie diese Zeilen als Unterroutine. Verwenden Sie Ihren Editor und ändern Sie Zeile 40 in 200 und Zeile 50 in 300. Fügen Sie nun einen Doppelpunkt und ein RETURN an Zeile 200 und 300. Nun ändern Sie Zeile 40, 60 und 80 so, daß Sie GOSUB 200, und Zeile 50 und 70 so, daß Sie GOSUB 300 lesen können. Weiter brauchen wir noch die Zeile 110 END. Das Programm sollte nun folgendermaßen aussehen:

```
10 PRINT CHR$(147)
20 PRINT "<SHIFT-U> <7MAL SHIFT-D> <SHIFT-I>
30 PRINT "<SHIFT-G> 2 <SHIFT-A> 5LEER 2 <SHIFT-H>"
40 GOSUB 200
50 GOSUB 300
60 GOSUB 200
70 GOSUB 300
80 GOSUB 200
90 PRINT "<SHIFT-G> 5LEER 2 <SHIFT-A> <SHIFT-H>"
100 PRINT "<SHIFT-J> <7MAL SHIFT-F> <SHIFT-K>"
110 END
```

```
200 PRINT "<SHIFT-G> 7LEER <SHIFT-H>":RETURN
300 PRINT "<SHIFT-G> 3LEER <SHIFT-A> 3LEER <SHIFT-H>":RETURN
```

Das hat uns zwar nicht viel Programmzeit eingespart, aber wenn Sie Bildschirmgrafik erst einmal wie ein normales Programm betrachten, werden Sie selbst nach Kürzungen suchen, um Speicherplatz und Laufzeit zu minimieren. Sehen wir nun, wie einfach es ist, aus der "Pik-2" eine "Herz-3" zu machen. Ändern Sie mit Hilfe des Editors die "Pik-2" in den Zeilen 30 und 90 in eine "Herz-3"- 3 (SHIFT-S), und das Pik in Zeile 300 in ein Herz. Ändern Sie weiterhin die Zeile 60 von GOSUB 200 in GOSUB 300. Wenn Sie jetzt RUN eingeben, haben Sie eine andere Karte mit nur wenigen Änderungen. Probieren Sie auch die anderen Farben und versuchen Sie so, ein vollständiges Kartenspiel zu erhalten.

Editieren!

Wenn Sie zum Ändern der Zeilen nicht Ihren Editor benutzt haben, hatten Sie hart daran zu arbeiten. Alles was dazu wirklich nötig gewesen wäre, ist jedoch nur die Eingabe der Änderung in der betreffenden Zeile und dann das Drücken der <RETURN>-Taste. Um eine Zeilennummer in eine andere zu ändern, überschreiben Sie nur die alte Zeilennummer mit der neuen. Um zum Beispiel die Zeile 40 in unserem Orginal "Pik-2" zur Zeile 200 zu machen, benutzen Sie die Cursortaste und gehen zur Zeile 40. Plazieren Sie den Cursor über die "4", geben Sie "200" ein und drücken <RETURN>. Wenn Sie das Programm jetzt auflisten, ist die Zeile 40 immer noch in der Originalform vorhanden, zusätzlich aber eine Zeile 200, die identisch mit Zeile 40 ist.

7.2 Bringen Sie Farbe in Ihre Grafik

Wenn alle Grafikzeichen nur in den zwei Blauschattierungen am Bildschirm dargestellt würden, wäre dies etwas langweilig. Wenn Sie jedoch keinen Farbbildschirm haben, erscheinen Ihre Grafikzeichen schwarz/weiß oder grün (wenn Sie einen grünen Monitor haben). Die verschiedenen Farbmuster erzeugen verschiedene Dichten in den Zeilen oder Figuren, die Sie entwerfen. Benutzen Sie keinen Farbmonitor, dann experimentieren Sie am besten mit weiß (CHR$(5) oder CTRL-2), wenn Sie auf die entsprechenden Kommandos stoßen. Wenn Sie später zur Mustererzeugung am Normalbildschirm kommen, können Sie diese mit den Farbcodes für verschiedene Effekte variieren.

Falls Sie einen Farbbildschirm haben, ist es gegebenenfalls notwendig, Ihren TV/Monitor anzupassen, damit Sie die korrekten Farben erhalten. Am besten tun Sie das, indem Sie ein kleines Farbtestprogramm laufen lassen. Das folgende Programm benutzt nur die Hälfte des Farbbereichs Ihres COMMODORE-64, da wir mit Hilfe der Tastaturgrafik und CHR$-Kommandos nur die Hälfte ausgeben können. Die zweite Hälfte sehen wir uns etwas später an. Im Moment reicht die Hälfte der Farbpalette aus, damit Sie Ihren Bildschirm einstellen können. (Lassen Sie die REM-Angaben aus, wenn Sie das Programm jetzt eingeben.)

```
10 PRINT CHR$(147)
12 FOR V = 1 TO 7:NEXT V:REM TABULATOR 7 ZEILEN
20 FOR I = 1 TO 8:READ C(I):NEXT:REM LIEST FARBCODES VOM
DATA-STATEMENT IN ZEILE 100
30 FOR J = 1 TO 8
40   PRINT  CHR$(18);CHR$(C(J));:FOR  K  =  1  TO   30:PRINT
CHR$(32);:NEXT K:REM SCHALTET FARBGRAPHIKMODUS EIN, WÄHLT
FARBE AUS UND DRUCKT 30 LEERSTELLEN
50 PRINT:NEXT J:PRINT CHR$(5)
60 PRINT SPC(10);"FARBTAFEL"
100 DATA 5,28,30,31,144,156,158,159:REM FARBCODES
```

Geben Sie RUN ein und passen Sie Ihren Monitor an. Wenn Sie das durchgeführt haben, beschäftigen wir uns mit den unterschiedlichen Farben.

Zurück zum Normalzustand

Wenn wir uns auf dem Bildschirm alle Farbnuancen ansehen möchten, müssen wir in den Normalzustand zurückkehren, indem wir gleichzeitig die Tasten <RUN/STOP> und <RESTORE> drücken.

Lassen Sie uns zu unserem "Pik-2"-Programm zurückkehren. Weil Pik schwarz ist, wechseln wir jetzt die Farbe unserer Karte von hellblau in schwarz. Dazu laden Sie bitte Ihr Programm in den Speicher and fügen Sie folgende Zeile hinzu:

```
15 PRINT CHR$(18);CHR$(144)
```

Das war sehr einfach. Machen Sie das gleiche bitte mit Ihrem "Herz-3"-Programm. Anstelle von CHR$(144) benutzen wir jedoch CHR$(28) für rot. Die folgende Aufstellung zeigt Ihnen die Farben und die dazugehörigen CHR$ Werte.

FARBE	CHR$ - WERT
WEISS	5
ROT	28
GRÜN	30
BLAU	31
SCHWARZ	144
PURPUR	156
GELB	158
CYAN	159

Lassen Sie uns nun ein einfaches Balkendiagramm mit Hilfe der Bildschirmgrafik anfertigen und diese mit Text am unteren Bildschirmrand beschriften.

```
10 PRINT CHR$(147)
20 INPUT "TITEL DER ZEICHNUNG";T$
30 PRINT:PRINT:INPUT "WIE VIELE BALKEN (1-7)";P%
32 IF P% > 7 THEN 10
40 FOR C = 1 TO P%:READ C(C):NECT C
50 FOR I = 1 TO P%
60  PRINT "DARZUSTELLENDER WERT FÜR BALKEN #";I;:INPUT  "(1-
20)";P(I):IF P(I) > 20 THEN 60
70 NEXT I
80 REM *** ENDE DES EINGABEBLOCKS ***
100 PRINT CHR$(147):S = 4
110 FOR I = 1 TO P%
120 PRINT CHR$(19)
130 FOR V = 0 TO (20-P(I)):PRINT:NEXT V
140 FOR PT = 1 TO P(I)
150 PRINT CHR$(18);CHR$(C(I));SPC(S);CHR$(32);CHR$(32)
155 NEXT PT
160 PRINT CHR$(5);TAB(S);I:S = S + 4
170  NEXT  I:PRINT  CHR$(28);:FOR  LN  =  1  TO  39:PRINT
CHR$(100);:NEXT LN
180 L = 20 - LEN(T$)/2:PRINT CHR$(5);SPC(L);T$
190 GET A$:IF A$ = "" THEN 190
200 REM *** ENDE AUSGABEBLOCK ***
500 DATA 5,28,30,144,156,158,159:REM ALLE FARBEN (OHNE BLAU)
```

Geben Sie RUN ein. Sie sehen, wie schön Sie Daten grafisch darstellen können. Das Programm ist limitiert, da Sie maximal 7 Balken ausgeben können, deren Werte zwischen 0 und 20 liegen müssen. Es ist jedoch einfach, die Anzahl der Balken zu ändern: Ersetzen

Sie den Wert durch einen höheren, ändern Sie die Anzahl der Farben, ändern Sie die Distanz (S-Variable) und rücken Sie die Balken näher zusammen durch Angabe eines CHR$(32) in Zeile 150. Liegen die Werte jedoch über 20, sind umfangreichere Manipulationen erforderlich, da die maximale Höhe des Balkens am Bildschirm 20 beträgt. Wir werden uns jetzt eine Übung mit zwei Balken ansehen, bei der Sie jeden Zahlenbereich eingeben können.

```
10 PRINT CHR$(147)
20 PRINT:PRINT:INPUT "MAX VALUE ->"";MV
30 N = 1:NN = MV:REM FÜR GRÖßERE GENAUIGKEIT KANN MAN N = .1
EINSETZEN
40 IF NN > 20 THEN N = N + 1:NN = MV / N:GOTO 40
50 FOR I = 1 TO 2
60 PRINT "DARZUSTELLENDER WERT";I;:INPUT PV(I)
70 PV(I) = INT(PV(I) / N)
80 NEXT I
90 REM *** ENDE DES EINGABEBLOCKS ***
100 PRINT CHR$(147);:FOR PL = 1 TO 2
110 C$ = CHR$(32) + CHR$(32) + CHR$(32) + CHR$(32):REM DIE
BALKEN WERDEN 4 SPALTEN BREIT
120 PRINT CHR$(19):FOR V = 0 TO (20-PV(PL)):PRINT:NEXT
130 FOR PT = 1 TO PV(PL):PRINT CHR$(18),CHR$(28);SPC(PL*10);
C$:NEXT PT
140 NEXT PL
150 FOR LN = 1 TO 39:PRINT CHR$(30);CHR$(100);:NEXT
160 PRINT CHR$(5)
170 PRINT SPC(9);"BLKN 1 ";SPC(5),"BLKN 2";
180 GET A$:IF A$ = "" THEN 180
```

Um zu zeigen, was hier vor sich geht, werde ich die entsprechenden Zeilen durchgehen und jede einzeln erklären.

1. In Zeile 30 wird eine Variable NN definiert. Sie erhält den maximalen Wert von MV, der in Zeile 20 eingegeben wurde.

2. In Zeile 40 wird NN mit 20 verglichen, um zu sehen, ob der maximale Wert größer 20 ist. Dies dient der Erzeugung einer proportionalen Skala. Ist der Wert größer 20, wird zur Variablen N 1 dazuaddiert, NN wird neu definiert und stellt den Wert von MV, diviert durch N dar. Es findet anschließend ein Sprung zum Anfang der Zeile statt; ein neuer Vergleich wird durchgeführt. So-

bald die Bedingung NN > 20 nicht mehr zutrifft (MV / N), wird die Schleife beendet. Der Wert von N wird im weiteren Programmverlauf dazu benutzt, um jeden eingegebenen Wert durch den Wert von N zu dividieren.

ZUM BEISPIEL:

Der Wert von MV wird auf 100 festgesetzt. Da 100 größer 20 ist, wird 1 zu N addiert und 100 jetzt durch 2 dividiert. Das Ergebnis 50 wird in der Variablen NN gespeichert. Da 50 immer noch größer 20 ist, erhöht sich N auf 3. Wenn MV jetzt durch 3 dividiert wird, erhalten wir den Wert 33,33. Wieder ist dieser größer als 20 und es findet ein weiterer Schleifendurchlauf statt. Die Schleife wiederholt sich bis N den Wert 5 hat. MV, dividiert durch N, ergibt jetzt den Wert 20. Wenn nun ein Vergleich mit 20 durchgeführt wird, steht fest, daß NN nicht größer als 20 ist und die Schleife wird mit dem Wert 5 für N verlassen. Ganz gleich, welchen Wert Sie eingeben: solange Sie nicht den maximalen Wert ausführen, tritt kein Fehler auf, da alle Balkenwerte wie PV(1) usw. durch 5 dividiert werden. Da 100 der eingegebene maximale Wert ist, stellt 20 den maximal auszugebenden Wert dar.

3. Für PV(I) werden in Zeile 60 zwei Werte eingegeben und diese Werte in Zeile 70 durch N dividiert. Das INT-Kommando dient zur Erzeugung einer Ganzzahl (Integer) für den Balken.

4. In Zeile 110 wird C$ für die Verkettung von vier Leerzeichen, CHR$(32), definiert. Dies dient dazu, um die Balken in der Breite von vier Zeichen darzustellen.

5. Zeile 120 bis 140 zeichnen die Balken wie in unserem ersten Balkendiagrammprogramm.

Für Perfektionisten mit etwas Zeit

Jedesmal, wenn wir Zeile 40 ausführen, erhöhen wir N um 1. Wünschen wir einen etwas geringeren Wert, können wir N auch um .1 oder .01 oder sogar .00001 erhöhen. Dies liefert uns einen angenäherten Minimalwert, durch den PV(I) dividiert wird, und doch erreichen wir damit eine Proportion. Es dauert jedoch auch länger, den Minmalwert von N in der Schleife zu finden. Ändern Sie das Programm ab, um die unterschiedlichen Ergebnisse zu sehen. Je kleiner die Erhöhung, desto näher erscheint der maximale Wert an der Obergrenze der Ausgabe, und desto länger braucht das Programm zur Ausführung.

Wir haben uns nun sehr lange mit der Ausgabe von Balken mit Hilfe der Bildschirmgrafik beschäfigt. Es ist jedoch ebenso wichtig, sich die praktischen Anwendungsmöglichkeiten dieser Grafikfunktion anzusehen. Die Benutzer sehen Bildschirmgrafik oft nur als Ausgabemöglichkeit für Mosaikbilder auf dem Bildschirm an. Aber wie wir gesehen haben, ist es möglich, diese Funktion sehr gut praktisch einzusetzen. Lassen Sie uns noch etwas Bewegung auf den Bildschirm bringen, bevor wir dazu übergehen, Grafik auf den Bildschirm zu poken.

Bewegung auf dem Bildschirm kann für Spiele und für spezielle Effekte genutzt werden. Wir geben Ihnen einige grundlegende Beispiele, um Ihnen das Konzept zu zeigen. Durch das Plazieren eines Zeichens auf dem Monitor, dessen Aufnehmen und erneutes Plazieren auf einer anderen Position, erzeugen Sie die Illusion von sich bewegenden Bildern. Das funktioniert auf die gleiche Weise wie ein Zeichentrickfilm. Eine Reihe von Bildern blitzt in gleichmäßigen Abständen am Bildschirm auf. Auch wenn jedes einzelne Bild eine gleichbleibende Figur darstellt, scheinen sich diese Figuren durch das rasche Aufblitzen in einer Folge zu bewegen. Ihr Computer macht das gleiche. Das folgende kleine Programm läßt zum Beispiel einen Ball in der linken oberen Ecke des Schirms aufhüpfen.

```
10 PRINT CHR$(147)
20   PRINT "<SHIFT-Q>  ":REM VOR DEM LETZTEN HOCHKOMMA
```

```
LEERZEICHEN!
30 FOR I = 1 TO 100:NEXT
40  PRINT  CHR$(19):PRINT " <SHIFT-Q>":REM NACH  DEM  ERSTEN
HOCHKOMMA LEERZEICHEN!
50 FOR I = 1 TO 100:NEXT
60 PRINT CHR$(19):GOTO 20
```

Was als hüpfender Ball erscheint, ist in Wirklichkeit eine auf dem Bildschirm plazierte Figur, die gelöscht und an einer anderen Position wieder angezeigt wird. Machen wir das gleiche nun in der vertikalen Achse mit Hilfe der Cursorsteuerung im Programm. Zum Spaß fügen wir noch Töne und andere spezielle Effekte hinzu.

```
10 PRINT CHR$(147):REM *** BEGINN DES GRAPHIKBLOCKS ***
20 FOR I = 1 TO 25
30  PRINT  TAB(20);"<SHIFT-Q><CURSOR HOCH>":REM EIN  WEIßER,
INVERSER BALL ERSCHEINT AM BILDSCHIRM
40 FOR J = 1 TO 50:NEXT J:REM VERZÖGERUNG
50 PRINT TAB(20); "<SPACE>":REM LÖSCHT DEN BALL
60 PRINT
70 NEXT I
80 GOSUB 200
90 PRINT TAB(20);"*":REM ***ENDE DES GRAPHIKBLOCKS ***
100 GET A$:IF A$ = "" THEN 100
110 END
200 REM *** TÖNE ***
210 POKE 54296,15:POKE 54277,20:POKE 54278,60:REM LAUTSTÄRKE
, ATTAKC/DECAY, SUSTAIN/RELEASE
220 POKE 54273,5:POKE 54272,185:POKE 54276,129:REM FERQUENZ,
WELLENFORM
230 FOR DU = 1 TO 50:NEXT DU:REM DAUER
240 POKE 54277,0:POKE 54278,0:POKE 54276,0:REM TONERZEUGUNG
AUS
```

Durch das Experimentieren mit verschiedenen Algorithmen können Sie eine Menge von Effekten erzielen. Wenn Sie Arcadespiele mit Bewegung und Tönen kennen, haben Sie nun eine Vorstellung davon, wie diese programmiert wurden. Beginnen Sie also an diesem SUPER-WELTRAUM-MONSTER Spiel zu arbeiten.

Da wir jetzt wissen, wie sich Figuren am Bildschirm bewegen, lassen Sie uns tiefer in die Anwendung der Farbe einsteigen. Das erste, was wir uns ansehen, ist die Änderung der Hintergrundfarbe und der Umrahmung des Bildschirms. Später betrachten wir die

Speicherbereiche für die Farbe, um alles an jede Stelle in jeder Farbe auf den Bildschirm zu bekommen.

Zu Beginn sehen wir uns noch einmal unser "Pik-2"-Programm an. Wir wissen schon, daß die Karten schwarz und nicht hellblau sein sollten. Jeder Kartenspieler weiß aber auch, daß "grüner Filz" den korrekten Hintergrund des "Tisches" bildet. Um den Hintergrund und die Umrandung des Bildschirms zu verändern, benutzen wir folgende POKEs:

```
POKE 53281,(0-15) HINTERGRUNDFARBE
POKE 5328,(0-15) FARBE DES BILDSCHIRMRANDES
```

Laden Sie "Pik-2" in Ihren Speicher, drücken CTRL-1, um die Zeichenfarbe Schwarz zu erhalten und geben Sie POKE 53281,5 ein. Wenn Sie das Programm jetzt laufen lassen, bekommen Sie eine schwarze Karte auf grünem Hintergrund. Da die Umrandung des Tisches meist aus Holz ist, gestalten wir die Umrandung des Bildschirms mit einem POKE 53280,9 braun. Unser Bildschirm sieht jetzt mehr wie ein Spieltisch aus. (Um den Normalzustand zu bekommen, drücken Sie die Tasten <RUN/STOP> und <RESTORE>.)

Es ist sehr einfach, die Farbe des Hintergrunds und der Umrandung zu ändern. Die folgende Liste zeigt Ihnen den Code für die 16 möglichen Farben.

CODE	FARBE	CODE	FARBE
0	SCHWARZ	9	BRAUN
1	WEISS	10	HELLROT
2	ROT	11	GRAUSTUFE 1
3	CYAN	12	GRAUSTUFE 2
4	VIOLETT	13	HELLGRÜN
5	GRÜN	14	HELLBLAU
6	BLAU	15	GRAUSTUFE 3
7	BELB		
8	ORANGE		

Das folgende kleine Programm führt Sie kurz durch die verschiedenen Hintergrund- und Randfarben.

```
10 PRINT CHR$(147)
20 BG = 53281:BD = 53280
30 INPUT "HINTERGRUNDFARBE ";B1:IF B1 > 15 THEN 30
```

```
40 INPUT "RANDFARBE        ";B2:IF B2 > 15 THEN 40
50 POKE BG,B1:POKE BD,B2
60 GOTO 10
```

Wenn Sie RUN eingeben, können Sie mit verschiedenen Textfarben experimentieren durch Drücken von CTRL und einer Taste von 1 bis 8. Sie werden sehen, daß bestimmte Textfarben auf bestimmten Hintergrundfarben mehr oder weniger deutlich sind. (Schwarz auf schwarz ist sehr schwer zu lesen!)

Sie haben vielleicht bemerkt, daß wir 16 verschiedene Farben für den Hintergrund und den Rand des Bildschirms festlegen können. Für unsere Tastaturgrafik haben wir jedoch nur 8 Farben. Um die Zeichen der Tastatur in allen Farben darzustellen, müssen wir die Speicherplätze des COMMODORE-64 für den Bildschirm und die Farbe genau kennen. Wie Sie bereits wissen, ist unser Bildschirm eine Matrix von 40 x 25. Jedes Element dieser Matrix steht für eine bestimmte Adresse im Hauptspeicher. Der Bildschirmspeicherbereich beginnt bei Adresse 1024 und endet bei 2023. Er stellt somit 1000 Positionen bereit, um etwas auf dem Bildschirm darzustellen. Durch das Poken verschiedener Werte auf diese Positionen, können Sie Zeichen in jeder beliebigen Lage am Bildschirm darstellen. Jedes Zeichen hat einen Code (ähnlich dem CHR$-Code); allerdings wird der Code gepokt und nicht geprintet. Der Code für "A" ist zum Beispiel 1. Wenn Sie mit POKE eine Speicherstelle zwischen 1024 und 2023 mit "1" belegen, erscheint auf der festgelegten Stelle ein "A". Ein Beispiel: löschen Sie Ihren Bildschirm, und geben Sie POKE 1475,1 ein. In der Mitte Ihres Bildschirms erscheint jetzt ein weißes "A". Beachten Sie auch die Position des Cursors: er steht noch immer oben am Bildschirm, weil Sie eine Speicherstelle dargestellt und nicht ein Zeichen mit PRINT ausgegeben haben. Um die Speicherplätze genauer zu betrachten, füllen wir diese mit "A's" auf, indem wir folgendes im Kommandomodus eingeben:

```
FOR I = 1024 TO 2023:POKE I,1:NEXT
```

Das folgende kleine Programm durchläuft alle Codes und stellt sie in der Mitte des Bildschirms dar.

```
10 PRINT CHR$(147)
20 FOR I = 0 TI 127
30 POKE 1484,I
40 FOR J = 1 TO 200:NEXT J:NEXT I
```

Machen wir eine kleine Wanderung über unseren Bildschirm. Wir starten mit der Speicherstelle 1024 und wandern mit einem Pfeil (Code 62) zur Speicherstelle 2023.

```
10 PRINT CHR$(147)
20 BG = 1024:ES = 2023
30  FOR  I = BG TO ES:POKE I,62:FOR J = 1 TO  5:NEXT  J:POKE
I,96
40 FOR K = 1 TO 5:NEXT K:NEXT I
```

Um die Bewegung am Bildschirm darzustellen, poken wir erst einen Pfeil (Code 92) und anschließend eine Leerstelle (Code 96) in die gleiche Speicherstelle, belassen jedes Zeichen kurz an der Stelle, gehen dann zur nächsten Speicherstelle und wiederholen den Vorgang. Durch die Benutzung der Bildschirmspeicherplätze haben wir eine bessere Kontrolle über die Zeichen und Bewegungen und können von einem Punkt zum anderen springen, ohne uns um die Position des Cursors zu kümmern. Kehren wir zurück zu unserem hüpfenden Ball; wir verwenden diesmal jedoch POKEs.

```
10 PRINT CHR$(147)
20 FOR I = (1024 + 20) TO 1984 STEP 40
30 POKE I,81:FOR D = 1 TO 10:NEXT D
40 POKE I,96:FOR X = 1 TO 10:NEXT X:NEXT I
50 FOR I = (1984 + 20) TO 1024 STEP -40
60 POKE I,81:FOR D = 1 TO 10:NEXT D
70 POKE I,96:FOR X = 1 TO 10:NEXT X:NEXT I
80 GOTO 20
```

Beachten Sie, daß wir an der linken oberen Ecke angefangen und eine Distanz von "20" dazuaddiert haben, um den Ball in die Mitte des Bildschirms zu bringen. Ab Zeile 50 führen wir den Prozeß in umgekehrter Richtung durch.

Lassen Sie uns die Speicherbereiche für die Farbe näher ansehen. Sie beginnen auf Position 55296 und enden bei 56295. Wiederum sind es 1000 Speicherplätze; betrachten Sie diese als Überlagerung der Bildschirmbereiche. Die erste Stelle des Bildschirmbereichs ist 1024; für den Farbbereich ist es 55296. Zunächst poken wir ein Zeichen in den Bildschirmspeicherplatz und anschließend eine Farbe für dieses Zeichen in den Farbspeicherplatz.

```
POKE 1024,81 <RETURN>
POKE 55296,8 <RETURN>
```

Als erstes erscheint der Ball, der dann die Farbe zu orange ändert, einer Farbe, die Sie vorher für die Zeichen nicht zur Verfügung hatten.

Nun sind Sie vielleicht an einem Punkt, wo Sie sich fragen: "Wie um alles in der Welt soll ich die entsprechende Stelle unter den Tausenden herausfinden und außerdem einen der 127 Zeichencodes. Und wie kann ich jetzt noch tausend verschiedene Farbspeicherplätze darüberlegen, mit einer der 16 Farben des Bildschirmspeicherbereiches und dann das ganze noch auf einen bestimmten Platz bringen?" In Wirklichkeit ist dies aber nicht so schwierig, wie es sich anhört. Übergeben Sie diese Arbeit doch Ihrem Computer, wie alle anderen Berechnungen auch! Das folgende ist Schritt für Schritt ein Überblick, den wir für ein Programm festlegen, das diese Berechnungen mit Variablen durchführt.

```
BS = 1024        <- BEGINN DES BILDSCHIRMSPEICHERS

BC = 55296       <- BEGINN DES FARBSPEICHERS

CS = XXXX        <- AUGENBLICKLICHE POSITION DES ZEICHENS

OF = CS - BS     <- DISTANZ ZWISCHEN ZEICHENPOSITION UND
                    BEGINN DES BILDSCHIRMSPEICHERS

CC = BC + OF     <- PLATZ DER FARBE

C1 = XX          <- ZEICHENCODE 0 - 127

C2 = XX          <- FARBCODE    0 - 15
```

Grundsätzlich bestimmen Sie die korrespondierende Speicherstelle des Bildschirm- und Farbspeichers, indem Sie einfach die Speicherstellen vom Anfang des Bereichs bis zu dem betreffenden Speicherplatz zählen. Da beide Bereiche fortlaufende Adressen benutzen, können wir für beide Bereiche auch die gleiche Distanz verwenden. Das folgende Programm benutzt die obigen Variablen und ermöglicht Ihnen die Plazierung eines Zeichens auf einer beliebigen Position.

```
10 PRINT CHR$(147)
20 BS = 1024:BC = 55296
30 INPUT "KOORDINATE  IM BILDSCHIRMSPEICHER (1024-2023)";C$
40 INPUT "CODE DES DARZUSTELLENDEN ZEICHENS (   0- 127)";C1
```

```
50 OF = CS - BS:CC = BC + OF
60 INPUT "FARBCODE                          (   0-  15)";C2
70 POKE CS,C1:POKE CC,C2
80 GET A$:IF A$ = "" THEN 80
90 GOTO 10
```

Spielen Sie mit dem Programm, bis Sie wissen, welcher Code die verschiedenen Zeichen und Farben liefert. Anschließend probieren Sie das folgende Programm aus, welches Ihnen einen "Perlenschleier" und einen anderen Weg zur Erzeugung von Farbeffekten unter Verwendung von programmierten POKEs zeigt.

```
10 PRINT CHR$(147)
20 BS = 1024:ES = 2023
30 BC = 55296:EC = 56295
40 FOR I = BS TO ES:POKE I,81:NEXT I
50 FOR C = BC TO EC STEP 2:POKE C,8:NEXT C
60 FOR NC = (BC + 1) TO EC STEP 2:POKE NC,4:NEXT NC
70 FOR C = BC TO EC STEP 2:POKE C,5:NEXT C
80 FOR NC = (BC + 1) TO EC STEP 2:POKE NC,6:NEXT NC
90 GET S$:IF S$ = "" THEN 90
100 PRINT CHR$(147)
```

Stellen Sie sich vor, Sie möchten nicht nach jedem Zeichen suchen, das Sie eingegeben haben. Angenommen, Sie möchten Ihren Namen oder eine Überschrift mit Hilfe der INPUT-Anweisung und der Tastatur schreiben, dann können Sie jetzt die Bildschirmbereiche benutzen und Zeichen hineinpoken. Während wir dies machen, lernen wir ein neues Kommando kennen, ASC. Das ASC-Kommando wandelt das erste Zeichen eines Strings in einen CHR$-Code. Zum Beispiel ist der ASCCI Code für ein "A" 65. Wenn Sie nun

```
PRINT ASC("A")
```

eingeben, bekommen Sie

```
65.
```

Da Sie diesen CHR$-Wert zum Poken schlecht nehmen können, wandeln Sie diesen CHR$-Code in einen zu pokenden Wert um. Wenn Sie sich nun die Bildschirmausgabe, die ASCII- und CHR$-Codes ansehen, stellen Sie fest, daß das ASCII-Alphabet bei 65, der Bildschirmcode aber bei 1 beginnt. Um eins ins andere zu verwandeln, brauchen wir nur einfach 64 zu addieren oder zu subtrahieren, je

nachdem, in welche Richtung wir konvertieren. Da wir ASCII in Bildschirmcode konvertieren möchten, subtrahieren wir 64 vom jeweiligen ASCII Wert (ASC). Tippen Sie folgendes ein:

```
10 PRINT CHR$(147)
20 INPUT "GEBEN SIE EINEN BUCHSTABEN (A-Z) EIN:"A$
30 A = ASC(A$):A1 = A - 64
40 POKE 1024,A1
50 GOTO 20
```

Wie Sie sehen, erscheint jetzt jedesmal, wenn Sie ein Zeichen eingeben, dieses in Weiß am oberen linken Bildschirmrand (Stelle 1024).

Da Sie nun wissen, wie Sie einzelne Buchstaben auf einer bestimmten Stelle des Bildschirms ausgeben können, versuchen Sie das gleiche mit längeren Strings. Um dies zu erreichen, machen Sie bitte folgendes:

1. Definieren Sie einen String oder geben Sie diesen mit INPUT ein.

2. Teilen Sie den String in einzelne Buchstaben, so daß Sie für jedes Zeichen seinen ASCII-Wert ermitteln können. (Das ASC Kommando liest nur das ganz links stehende Zeichen des Strings.)

3. Konvertieren Sie den ASCII-Wert in den Bildschirmcode.

4. Poken Sie den ermittelten Code.

Wir benutzen die Distanz 64 und unser MID$-Kommando, um jedes Zeichen zu untersuchen. Wenn wir jedoch auf ein Leerzeichen stossen (ASCII Wert 32), bekommen wir Schwierigkeiten, da 32 - 64 eine negative Zahl ergibt. Um das zu vermeiden, führen wir eine Prüfung auf Leerstellen durch und definieren dieses mit dem korrekten POKE-Wert, der ebenfalls 32 ist. Um die Sache einfach zu gestalten, geben wir unsere Strings am oberen linken Bildschirmrand aus.

```
10 PRINT CHR$(147)
20 PRINT:PRINT:INPUT "IHR NAME:";NA$
30 BS = 1024:BC = 55296
40  DIM A(LEN(NA$)),A1(LEN(NA$)):REM DIMENSIONIERUNG,  FALLS
NAME LÄNGER ALS 11 ZEICHEN IST
50 FOR I = 1 TO LEN(NA$)
55 IF ASC(MID$(NA$,I,1)) = 32 THEN A1(I) = 32:NEXT I:REM
PRÜFUNG AUF LEERSTELLEN
60 A(I) = ASC(MID$(NA$,I,1)):A1(I) = A(I) - 64:REM UMWANDELN
DES ASCII-CODES IN BILDSCHIRM-CODE
70 NEXT I
80 FOR P = 1 TO LEN(NA$):POKE BS + P,A1(P):NEXT P:REM POKEN
100 REM *** FARBPALETTE ***
110 INPUT "FARBCODE (0-15)";C2
120 FOR C = 1 TO LEN(NA$):POKE BC + C,C2:NEXT C
130 GET W$:IF W$ = "" THEN 130
140 RUN
```

Wenn Sie jetzt aber einen Punkt (.) eingeben, lösen Sie einen ILLEGAL QUANTITY ERROR IN 60 aus, weil der Punkt, genau wie das Leerzeichen, einen kleineren ASCII-Wert als 65 hat. Wenn Sie auch Punkte mit eingeben möchten, werfen Sie einen Blick auf die Leerzeichenprüfung in Zeile 55. Versuchen Sie eine ähnliche Abfrage für Punkte einzubauen. Sie sollten jetzt allmählich wissen, wie Tastaturzeichen gepokt werden und wie Sie Programme schreiben können, die Strings in den Bildschirmspeicherbereich poken.

7.3 Spritegrafik

Lassen Sie uns jetzt noch eine weitere sehr leistungsfähige Möglichkeit des COMMODORE-64, die Spritegrafik, ansehen. Als erstes müssen wir erkären, was ein "Sprite" ist und wofür man es verwendet. Im Prinzip ist ein Sprite eine im Speicher hinterlegte Figur. Je nachdem, was Sie an bestimmten Speicherstellen hinterlegen, bekommen Sie verschiedene Figuren oder "Pixel". Sie werden in Spielprogrammen benutzt, dienen aber auch zur Belebung jeder anderen Darstellung.

Der Vorteil der Spritegrafik beim COMMODORE-64 liegt darin, daß Sie eine große Kontrolle bei der Erstellung haben, da Sie Sprites in einem verschlüsselten Binär-Code eingeben. Der Nachteil ist, daß die Spritegrafik ein bißchen schwieriger zu verstehen ist. Wenn Sie jedoch selbst Sprites zwischen die BASIC-Elemente des Programms einfügen können, lohnt sich die Anstrengung. Um Ihr Interesse zu wecken, beginnen wir mit einem einfachen Beispiel: Ein "Raumschiff". Wir erklären später alles. Geben Sie vorerst das Programm ein und sehen Sie sich an, was passiert:

```
10 PRINT CHR$(147)
20 BG = 53248:REM BEGINN DES GRAPHIK-CHIPS
30 TU = BG + 21:REM ADRESSE FÜR PIXEL 'XX'
40 S1 = 2:REM PIXELFAKTOR 1
50 L = 2040:REM ANFANGSADRESSE DES PIXELPOINTERS
60 22 = 832:REM ANFANGSADRESSE DER DATEN
70 SL = 13:REM DATENBLOCK NUMMER 13
80 X1 = BG + 2:Y1 = BG + 3:REM X,Y KOORDINATEN
90 REM *** ENDE DER VARIABLEN-DEFINITION
200 FOR R = 0 TO 29:READ D:POKE SS + R,D:NEXT R:REM EINLESEN
DER  DATEN  FÜR DIE KONSTRUKTION DER GRAPHIK IN  DEN  ZEILEN
500-520
210 POKE TU,S1:REM EINSCHALTEN PIXEL #1
220 POKE L + 1,SL:REM EINLESEN DER DATEN FÜR PIXEL #1
230 POKE Y1,100:REM VERTIKALE ADRESSE
240 FOR H = 0 TO 255
250 POKE X1,H:NEXT H:REM HORIZONTALE ADRESSE
260 GOTO 240: REM SCHLEIFE
270 REM *** ENDE GRAPHIKERSTELLUNG UND BEWEGUNG ***
500 DATA 2,0,64,2,0,64,2,24,64
510 DATA 2,60,64,2,102,64,3,255,192
520 DATA 2,60,64,2,24,64,2,0,64,2,0,64
```

Wenn Sie alles korrekt eingegeben haben, sollten Sie ein "Raumschiff" horizontal über den Bildschirm fliegen sehen. Es gibt noch eine Menge anderer Möglichkeiten mit Pixel; lassen Sie uns aber das Beispiel von oben verwenden, um zu erklären, was da vor sich geht.

Als erstes befassen wir uns kurz mit dem Konzept der Binärarithmetik. Wenn wir unsere Sprites als Punkte am Bildschirm betrachten die in Blöcke zusammengefaßt sind, verstehen wir sowohl die Binämathematik als auch die Sprites. Zu Beginn sehen wir uns die 8 Bits eines Bytes an, die von 7 bis 0 numeriert sind. Sie enthalten entweder eine 0 oder eine 1, die einzigen Ziffern des Binärsystems.

7	6	5	4	3	2	1	0
0	1	0	1	1	0	1	0

Für den Computer müssen alle Angaben im Binärsystem vorhanden sein. Der 6510 Microprozessor führt dies in Bereichen durch, die "Bytes" genannt werden. Jedes Byte ist 8 "Bits" lang. Die oben binär dargestellte Zahl "01011010" bedeutet die dezimale Zahl "90". Immer wenn Sie die Zahl "90" über die Tastatur eingeben, wird diese vom Computer in "01011010" übersetzt.

Er führt diese Konvertierung automatisch durch. Für die Spritegrafik müssen Sie diese Aufgabe selbst übernehmen. Es ist jedoch sehr einfach. Da wir in der Binärarithmetik nur zwei mögliche Zustände haben, hängen wir einfach eine 0 an, damit wir eine andere Kombinationsmöglichkeit erreichen. Das gleiche machen wir ja auch in der Dezimalarithmetik. Wenn wir z.B. die 9 erreichen, starten wir erneut mit "1" und hängen eine "0" an, um eine 10 zu bekommen. Wenn wir binär zählen, sieht das folgendermaßen aus:

0 = 0
1 = 1
10 = 2
11 = 3
100 = 4
101 = 5

110 = 6
111 = 7
1000 = 8

Es ist genau wie bei den Grenzen 9, 99 oder 999 im Zehnersystem: Wir starten mit 1 und hängen eine weitere Stelle an. Beim Binärsystem führen wir das gleiche bei 11, 111 oder 1111 durch, da wir nur die Zustände 0 und 1 zur Verfügung haben. Da wir nicht gewohnt sind, im Binärsystem zu arbeiten, brauchen wir einen einfachen Weg, um binär in dezimal zu konvertieren. Auf Seite 78 Ihres COMMODORE-64-HANDBUCHES finden Sie eine kleine Tabelle zur Umwandlung von der Binärdarstellung in die Dezimaldarstellung. (Geben Sie bei dem aufgeführten Programm noch folgende Zeile ein, um ein Abbruchskriterium für das Programm zu definieren: 11 IF A$ = "x" THEN END:REM BEENDEN DES PROGRAMMS DURCH EINGABE VON X.)

Eine andere einfache Art, Konvertierungen auszuführen, beruht auf der Tatsache, daß jedes Bit nur die Werte 0 oder 1 enthalten kann. Sehen wir uns dies genauer an:

WERT	128	64	32	16	8	4	2	1
BIT	7	6	5	4	3	2	1	0
	0	1	0	1	1	0	1	0

0 + 64 + 0 + 16 + 8 + 0 + 2 + 0 = 90

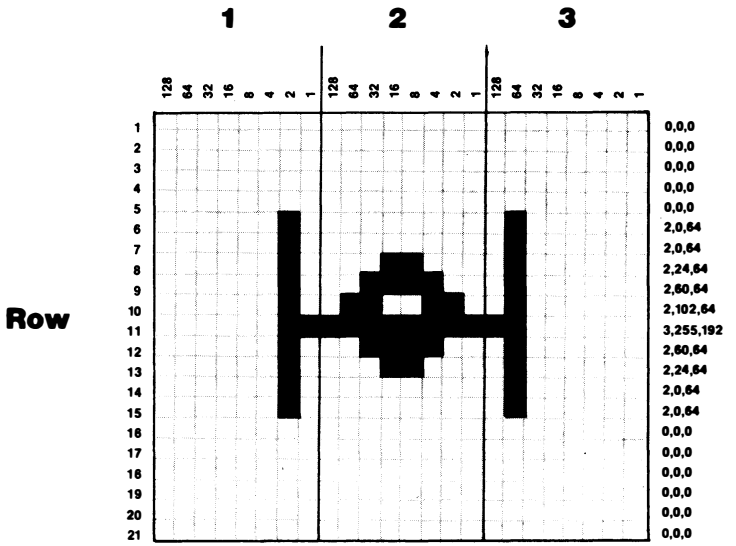
Um unsere dezimale Zahl zu ermitteln, brauchen Sie nur die Zahlen in der Reihe "WERT" zu addieren.

Eine naheliegende Frage ist: "Lohnt es sich, sich so zu quälen?" Nun, da Sprites aus kleinen Punkten bestehen, die durch die verschiedenen Bits ("on" oder "off") erzeugt werden, ist dies notwendig. Sie können aber auf diese Art jede beliebige Form erzeugen, indem Sie die Bits ein- oder ausschalten. Durch die Konvertierung dieser Binärmuster in dezimale Werte, können wir diese vom BASIC aus poken. Sehen wir uns zu Beginn an, wie unser "Raumschiff" erzeugt wurde. Hier ist alles, was Sie dazu brauchen:

1. Kariertes Papier.
2. Bleistift und Radiergummi.
3. Ein Lineal.

Zeichnen Sie auf das Papier eine Matrix, 24 Karos breit und 21 hoch. Teilen Sie die Matrix vertikal in drei gleiche Spalten. Jede Spalte ist nun 8 Karos breit und 21 hoch; jede Reihe besteht aus 3 Spalten von jeweils 8 Karos. Die 3 Spalten stellen 3 Bytes dar, genau wie in der Zeichnung im Buch. Da wir 21 Reihen und 3 Bytes pro Reihe haben, sind in unserem Block 21 x 3 oder 63 Bytes. Jeder Block stellt eine Grafik dar. Wir benutzen für das "Raumschiff" nicht jede Reihe, müssen den Block aber trotzdem als Gesamtheit betrachten. Wir lesen unsere Reihen von links nach rechts und stoppen zu Beginn jedes neuen Bytes. Wenn Sie das Konvertierungsprogramm des COMMODORE-64-HANDBUCHs benutzen, geben Sie die Zahlen des am weitesten links liegenden "Karos" oder "Bits" bis hin zu dem am weitesten rechts liegenden Bit des Bytes ein. VERWECHSELN SIE DIE EINZELNEN BITS NICHT!

Werfen wir einen Blick auf die folgende Figur, um zu sehen, wie die Spritegrafik entstanden ist.



Bei unserer Grafik geben wir in die ersten fünf Reihen keinerlei Information ein. Wir könnten für Reihe 1 - 5 genauso gut schreiben: DATA 0,0,0. Da wir unser "Raumschiff" aber in der Mitte des Blocks ausgeben möchten, gibt es keinen Grund dazu. (In einigen Anwendungen ist es jedoch erforderlich!) Ebenso enthalten die letzten sechs Reihen keinerlei Information. Wir können diese also leer lassen oder Nullen eingeben. Bei Anwendungen, in denen Sie mehrere Grafiken im gleichen Speicherbereich benutzen, müssen Sie den Block komplett füllen, um die unerwünschten Punkte der vorhergehenden Figur zu löschen. Sehen wir uns an einer Reihe an, wie ein Teil der Zeichnung konvertiert wird.

BYTE #1

WERT	128		64		32		16		8		4		2		1		
BITNR.	7		6		5		4		3		2		1		0		
	0		0		0		0		0		0		1		0		
	0	+	0	+	0	+	0	+	0	+	0	+	1	+	0	=	2

BYTE #2

WERT	128	64	32	16	8	4	2	1
BITNR.	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0

0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 0

BYTE #3

WERT	128	64	32	16	8	4	2	1
BITNR.	7	6	5	4	3	2	1	0
	0	1	0	0	0	0	0	0

0 + 64 + 0 + 0 + 0 + 0 + 0 + 0 = 64

Obwohl dieser Prozeß kompliziert und anspruchsvoll aussieht, ist er eher eine Frage der Organisation als der Komplexität. Vor allem ist das Zeichnen der Grafik auf Karopapier sehr einfach, und wenn Sie dann die detaillierte Grafik erhalten, ist das der Lohn für all Ihre Mühe. Wenn Sie Grafiken wie in unserem Beispiel erstellen, wiederholen sich auch oft Reihen. Wenn Sie erst die Werte dieser Reihen herausgefunden haben, brauchen Sie diese nur in die DATA-Zeile erneut eingeben.

Im nächsten Schritt werden wir Anweisungen für Tricks geben. Als erstes müssen wir uns über die Überlagerung des Bereichs, durch den wir uns bewegen, im klaren sein. Im Grunde ist es eine 320 x 200-Matrix. Wir können 320 Punkte oder Pixels horizontal und 200 vertikal bewegen. Ein Bereich ist jedoch 24 Pixels breit und 21 hoch. Wenn die Grafik fertiggestellt ist, wird der gesamte Bereich auf dem definierten Teil des Bildschirms ausgegeben und auf diesem fortbewegt. Um den Bereich über den Monitor zu bewegen, stellen wir uns den Bildschirm als ein Feld mit x- und y-Koordinaten vor. Die Position 160,100 stellt genau die Mitte dar. Der x-Wert bringt Sie nach rechts und der y-Wert nach unten.

Sichern Sie Ihre Grafik

Wenn Sie mit der Erstellung der Grafik fertig sind, wäre es gut, das Programm auf Diskette oder Band abzuspeichern. Indem Sie Zeilennummern größer 500 benutzen, brauchen Sie bei folgenden Programmen nur die DATA-Zeilen in den Speicher zu laden und anschließend die Parameter der darunterliegenden Zeilen zu definieren. Sie können damit Reihen der Grafik abändern oder neue Reihen hinzufügen. Sie müssen so nicht alle Grafiken erneut eingeben, sondern erstellen neue Grafiken durch die Modifizierung der alten oder durch Hinzufügen weiterer DATA-Zeilen.

Bevor wir zu den Bewegungen kommen, müssen wir unsere Bereichsdaten in den Speicher bringen und den Grafikbereich einrichten. In unserem Beispiel benutzen wir die Variable BG für den Beginn des Grafikbereichs, "Beginn Grafik" (BG = 53248). Die meisten anderen Speicherbereiche stellen eine Distanz zu BG dar. Wir benutzen folgende Variablen:

BG = 53248 - Beginn des Grafikbereichs

TU = BG + 21 - läßt einen Bereich erscheinen oder löscht ihn. TU steht für "Turn on" eines definierten Bereichs oder einer Reihe von Bereichen.

S1 = 2 - Wert der Bereichs #1. Bereichswerte sind:

S0 = 1
S1 = 2
S2 = 4
S3 = 8
S4 = 16
S5 = 32
S6 = 64
S7 = 128

L = 2040 - Distanz der Bereiche mit Angabe der Bereichsnummer, nicht des Wertes. Zum Beispiel: Bereich 0 = 0, Bereich 1 = 1, Bereich 2 = 2 usw. 2040 bis 2047 wird dazu benutzt, die Zeiger (Pointer) zu setzen, nicht zum Hinterlegen von Daten.

SS = 832 - Beginn des Bereichsspeichers. Hier werden die tatsächlichen Daten gespeichert. Es können 3 Gruppen von Datenspeichern benutzt werden. 832, 896(832 + 64) und 960 (895 + 64). Sie müssen getrennt gehalten werden und dürfen sich nicht überlagern, außer Sie möchten für den Bereich die doppelte oder dreifache Größe. Wir benutzen nur die erste Angabe.

SL = 13 - Speicher-"Block" (832/63) für den Datenbeginn bei 832. Für den Beginn bei 895 wird SL = 14, für 958 = 15 festgelegt.

X1 = BG + 2 : Y1 = BG + 3 - Die Werte stellen die x- und y-Koordinaten für den Bereich #1 dar. Alle x-/y-Paare beginnen für Bereich 0 mit 0,1; für Bereich 1 mit 2,3; für Bereich 2 mit 4,5 usw.

Um zu sehen, wie diese Variablen eingesetzt werden, gehen wir Schritt für Schritt durch das Programm und beginnen dabei mit Zeilennummer 200:

```
FOR R = 0 TO 29:READ D:POKE SS + R,D:NEXT R
```

Wir lesen hier unsere DATA-Anweisungen von 500-520 und hinterlegen die den einzelnen Werten entsprechenden Pixel in fortlaufenden Speicherbereichen, beginnend bei 832 (SS + 0).

```
POKE TU, S1
```

Dies aktiviert Bereich 1.

```
POKE L + 1, SL
```

Dies legt fest, daß die Daten von Block SL für Bereich #1 bestimmt sind.

POKE Y1, 100
Die y- oder vertikale Position soll sich an Stelle 100, in der Mitte des Bildschirms befinden.

FOR H = 0 TO 255

POKE X1, H: NEXT H
Dies bewegt den Bereich aus der horizontalen Position 0 nach 255. Die vertikale Position verbleibt auf 100; wir erreichen hiermit nur eine horizontale Bewegung.

Machen Sie sich keine Gedanken, wenn Sie im Moment noch nicht alles genau verstehen. Warum Informationen an bestimmten Speicherstellen und nicht an anderen gespeichert werden oder warum Blöcke gepokt werden, ist nicht das Hauptproblem. Beachten Sie die Reihenfolge der Kommandos und schauen Sie, was sich ereignet, wenn Sie Werte ändern. Welchen Wert müssen Sie z.B. verändern, wenn Sie den Sprite #3 anstelle des Sprites #1 aktivieren möchten? Wie können Sie erreichen, daß Ihr Sprite vertikal statt horizontal oder im Zickzack bewegt wird? Um dies herauszufinden, geben Sie Änderungen ein und wiederholen Sie die oben beschriebenen Variablen, bis Ihr Sprite die gewünschte Lage erreicht hat.

Farbe, Erweiterungen und die Kontrolle des gesamten Bildschirms in Verbindung mit der Spritegrafik

Da wir jetzt einiges über die Erstellung und Bewegung von Sprites wissen, sehen wir uns an, wie wir ihre Farbe verändern, sie erweitern und sie zu einer fortgesetzten Aktivität bringen können. Wir nehmen dazu unser Beispielprogramm und erweitern die vertikale sowie die horizontale Achse unseres Sprites. Wir benutzen folgende Variablen:

```
EX = BG + 29   Adresse der X-Achsenerweiterung
EY = BG + 23   Adresse der Y-achsenerweiterung
```

Fügen Sie folgendes zu Ihrem Programm:

```
85  EX = BG + 29: EY = BG + 23
225 POKE EX,S1: POKE EY,S1
```

Geben Sie jetzt RUN ein, bewegt sich Ihr Sprite in einer größeren Abmessung über den Bildschirm. Möchten Sie auch die Farbe ändern? Nun, dann fügen wir noch ein paar Variablen hinzu:

C = BG + 39	Beginn der Farbadresse
C1 = C + 1	Farbadresse für Sprite #1. Benutzen Sie die Spritenummer als Distanz (0 - 7).
POKE C1,2	Auswahl der Farbe Rot (2). Farbauswahl (0 - 15).

Um die Farbvariablen ins Programm zu integrieren, fügen Sie folgende Zeilen hinzu:

```
87  C = BG +39: C1 = C +1
227 POKE C1,2
```

Sie sollten jetzt ein großes rotes "Raumschiff" sehen, daß sich über den Bildschirm bewegt. Sie ändern die Farbe durch Eingabe der verschiedenen Werte, die in Zeile 227 nach C1 gepokt werden.

Als nächstes bewegen wir unser Sprite fortlaufend über den Bildschirm. Dies beinhaltet das "Aktivieren" (turn on) des "most significant" (bedeutendsten) Bytes (MSB). Sie sollten sich keine großen Gedanken darüber machen, was ein "MSB" ist, sondern dabei an einen "Bildschirmausbau" oder an das "rightmost screen movement" denken. Wir werden es zuerst "anschalten", unser Sprite bewegen, dann wieder "ausschalten" und zur Ausgangsposition zurückkehren. Das Ergebnis ist eine ununterbrochene Bewegung über die volle Bildschirmfläche.

RS = BG + 16 Rechte Bildschirmadresse

Jetzt möchten wir die letzten 64 Punkte auf die rechte Seite des Bildschirms bewegen und benutzen dazu eine Schleife von 0 bis 63. Fügen Sie folgende Zeilen hinzu:

```
88 RS = BG + 16
252 POKE RS,S1
254 FOR HH = 0 TO 63:POKE X1,HH:NEXT HH
256 POKE RS,0
```

Da wir nun die Tricks kennen, lassen Sie uns ein weiteres Sprite in unser Programm einfügen. Wir werden die beiden Sprites unabhängig voneinander bewegen und sie in verschiedenen Farben und unterschiedlichen Größen darstellen. Als erstes müssen wir ein weiteres Sprite zeichnen. Ich habe hier auch gleich noch ein anderes "Weltraum-Sprite" zur Hand. Fügen Sie zuerst die folgenden DATA-Zeilen hinzu:

```
600 DATA 0,8,0,0,28,0,3,255,224,6,127,48,12,62,24
610 DATA 24,28,12,112,8,7
```

Wir erstellen nun unser zweites Sprite, indem wir Sprite #4 modifizieren. Ändern Sie die folgenden Zeilen an Ihrem Programm:

```
40 S1 = 2:S4 = 16
80 X1 = BG + 2:Y1 = BG + 3:X4 = BG + 8:Y4 = BG + 9
87 C = BG + 39:C1 = C + 1:C4 = C + 4
205 FOR R = 0 TO 20:READ D:POKE (SS + 64) + R,D:NEXT R
210 POKE TU,S1 + S4
220 POKE L + 1,SL:POKE L + 4,SL + 1
227 POKE C1,13:POKE C4,10
245 POKE X4,H:POKE Y4,H
```

Das ist alles. Gehen wir das Programm Zeile für Zeile noch einmal gemeinsam durch:

Zeile 40: wir fügen den Wert (16) des Sprite #4 hinzu

Zeile 80: die x- und y-Werte für Sprite #4 werden hinzugefügt

Zeile 87: die Farbadresse für Sprite #4 wird definiert

Zeile 205: wir lesen die Daten für Sprite #4 ein und benutzen eine Distanz von 64, um die Daten in Block #14 zu speichern

Zeile 210: wir aktivieren (turn on) Sprite #1 und Sprite #4. (Beachten Sie, daß dies durch Aneinanderfügen der beiden Sprites erreicht wird, das ist besser als durch zwei separate POKEs.

Zeile 220: die Distanz für den Adressenpointer des zweiten Sprites wird gepokt und festgelegt, so daß diese im nächsten Block (SL + 1) zu finden ist.

Zeile 227: wir fügen die Farbe für Sprite #4 hinzu - hellrot, Color #10.

Zeile 245: die X- und Y-Position für Sprite #4 werden hinzugefügt.

Das war alles über die Sprites. Inwieweit Sie dieses neue Wissen nun selbst einsetzen, bleibt Ihnen überlassen. Die Anwendung ist etwas komplizierter als unsere bisherigen Übungen. Wenn Sie aber Ihre Programme in Blöcken organisieren, erkennbare Variablen einsetzen und die Reihenfolge einhalten, werden Sie doch eine Menge mit den Sprites erreichen. Mit Sprites ist die Belebung des Bildschirms viel einfacher als mit den vorher besprochenen Methoden. Wir müssen unsere Sprites nicht "löschen" und Sie haben vielleicht bemerkt, daß ein Sprite sich "hinter" dem anderen fortbewegen kann. Da sich Sprites gut bei Figuren einsetzen lassen, können Sie auch weitaus mehr Formen erzeugen als mit anderen Grafikzeichen.

7.4 Zusammenfassung

Das Kapitel führte uns durch die Welt der Computergrafik. Beginnend mit Bildschirmgrafik, haben wir gesehen, wie Sie Text und Grafik kombinieren können, um damit Bilder zu erzeugen. Weiter haben wir gelernt, wie wir Bilder auf dem Bildschirm bewegen können, indem wir sie in verschiedene Bildschirmbereiche poken, sie löschen, um sie dann an einer anderen Position wieder erscheinen zu lassen. Wir haben auch herausgefunden, wie wir unserer Grafik Farbe geben, entweder von der Tastatur aus oder durch Poken verschiedener Farbwerte. Durch die Programmierung von Distanzen konnten wir unsere Figuren und Farben koordinieren.

Der letzte Teil unserer Erforschung der Grafik führte uns in die Welt der Sprites. Ausgehend von einer Zeichnung auf Karopapier, haben wir unsere Kreationen durch Übersetzung der Figuren in den Binärcode in den Hauptspeicher des Computers übernommen. Weiter haben wir gelernt, wie wir die Informationen in den Speicher

bringen und ein sich bewegendes Sprite ausgeben können. Dann haben wir Farbe in das Ganze gebracht, das Sprite vergrößert und auf dem Bildschirm bewegt. Zum Schluß haben wir geübt, wie wir mehrere Sprites erzeugen und diese gleichzeitig bewegen können.

An dieser Stelle soll deutlich gemacht werden, daß sich Computergrafik nicht nur für Spaß und Spiel, sondern ebenso gut für praktische Anwendungen eignet. Wir haben zum Beispiel gesehen, wie man Grafiken mit den Grafiktasten erstellt, aber Sie können auch Spritegrafiken im Programm verwenden, um etwas deutlicher oder interessanter darzustellen. Wenn wir mit Spritegrafik und sich bewegenden Bildern arbeiten, denken wir natürlich zuerst an Spielprogramme. Setzen Sie sich jedoch bei der Anwendung von Grafiken hier keine Grenze. Versuchen Sie, diese zur Steigerung der Aussagekraft von Informationen einsetzen oder auf andere kreative Art zu nutzen.

8

Dateiverarbeitung

Einführung

In diesem Kapitel werden Sie mehr über fortgeschrittene Anwendungen mit dem Band- oder Diskettensystem lernen. Wir besprechen zwei Arten von Dateien: (1) Datendateien und (2) sequentielle Textdateien. Es gibt einige Ähnlichkeiten zwischen Daten- und sequentiellen (fortlaufend angelegt) Textdateien. Ihre Datendateien auf Diskette stellen eine Art sequentielle Datei dar, und wir können sogar sagen, daß die Methode, mit der Ihr Kassettengerät Daten speichert, eine Form von sequentieller Textdatei ist. Wir werden sie jedoch der Klarheit halber getrennt besprechen.

Bevor wir beginnen, möchte ich betonen, daß das COMMODORE-Diskettensystem sehr gut entwickelt ist. Für Anfänger kann es schwierig sein, einige der höherentwickelten Anwendungen zu verstehen. Außerdem besteht große Gefahr, Programme oder Daten auf der Diskette zu zerstören. Aus diesem Grund werde ich, trotz des Risikos, weitschweifig zu scheinen, die Abschnitte langsam durcharbeiten und die verschiedenen Funktionen der Kommandos für Ihr Diskettensystem erklären. Allerdings werden wir die komplizierteren Möglichkeiten des Disk-Operating-Systems nicht behandeln, da dies über die Zielsetzung dieses Buches hinausgehen würde. Ich empfehle Ihnen, eine leere formatierte Diskette für die Übung zu verwenden und Ihre Programmdisketten dadurch vor unbeabsichtigtem Löschen oder Überschreiben zu schützen.

8.1 Datendateien auf Band

Wäre es nicht vorteilhaft, wenn man große Datenmengen nach der Eingabe auf Band abspeichern könnte? Nehmen wir zum Beispiel an, Sie haben ein umfangreiches Sprite erstellt und möchten es in einem anderen Programm verwenden. Anstatt alles neu einzugeben, können Sie die Daten in jedes andere Programm laden. Nun, durch die Benutzung einer Datendatei können Sie dies und vieles andere durchführen. Sie können jede Art von Daten, numerische oder alphanumerische, auf Band sichern und mit Hilfe einer Reihe spezieller Kommandos, die Sie noch lernen werden, direkt in ein Programm laden. Sie können ein Scheckbuch-Programm erstellen, das alle Ihre Schecks und Salden auf Band sichert; eine Adressenliste

erstellen, die Namen, Adressen und Telefonnummern erzeugt, sichert und holt, und sogar eine Liste Ihrer Lieblingsrezepte anlegen.

In Kapitel 1 und 2 haben wir besprochen, wie Sie unter Verwendung der Commodore-C2N-Kassetteneinheit mit SAVE ein Programm abspeichern und es mit LOAD wieder in den Speicher des COMMODORE-64 bringen. Beide Kommandos werden im Kommandomodus ausgeführt. Die Kommandos, die wir jetzt besprechen, werden im Programmodus ausgeführt, auch wenn sie genauso Informationen auf Band abspeichern oder laden. Sie machen es in einem anderen Format. Zu Beginn wollen wir die unterschiedlichen Kommandos zur Bearbeitung von Datendateien wiederholen. Dann sehen wir uns dazu ein paar Programme an.

8.2 OPEN, INPUT#, PRINT# und CLOSE

Um Informationen von der Kassette zu lesen oder Daten aus dem Programm abzuspeichern, muß das Band durch einen OPEN-Befehl "vorbereitet" werden, der folgendes Format hat:

```
OPEN N,1,(0,1 or 2),"DATEINAME"
```

"N" kann jede Ganzzahl zwischen 1 und 255 sein, um eine Beziehung zur Datei herzustellen. Wenn Sie z.B. Ihre Datei mit der Zahl 21 ansprechen wollen, würden Sie schreiben:

```
OPEN 21, usw.
```

Da das Gerät als Kassettenrecorder angemeldet wird, ist die zweite Ziffer eine "1". Sie benutzen bei der Kassette immer die Ziffer "1", bei der Diskette immer "8".

```
OPEN 21,1, usw.
```

Ihre Datei ist mit einer "0" immer zum Lesen vorbereitet, mit einer "1" oder "2" zum Schreiben. Benutzen Sie die "1" für das Schreiben mit einer End-Of-File-Marke (Ende der Datei), die "2" für eine End-Of-Tape-Marke (Ende des Bands).

OPEN 21,1,0, etc.	Lesen einer Datei
OPEN 21,1,1, etc.	Schreiben einer Datei mit End-Of-File-Marke
OPEN 21,1,2, etc.	Schreiben einer Datei mit End-Of-Tape-Marke

Stellen Sie einen Namen für Ihre Datei bereit. Angenommen, Sie möchten Ihr erstelltes Sprite mit Daten einer Rakete abspeichern und nennen es "RAKETENSPRITE". Sie schreiben

```
OPEN 21,1,1,"RAKETENSPRITE"          oder

OPEN 21,1,2,"RAKETENSPRITE"
```

Um die Daten zu lesen, würden Sie eingeben

```
OPEN 21,1,0,"RAKETENSPRITE".
```

Dies mag kompliziert erscheinen; wenn Sie aber geübter sind, ist es sehr einfach. Zur gleichen Zeit sind Sie sehr flexibel, da es möglich ist, eine Anzahl verschiedener Dateien durch Vergabe verschiedener Namen zu eröffnen. Gewöhnlich schließen Sie eine Datei jedoch, bevor Sie eine andere eröffnen. Um eine Datei zu schließen, geben Sie CLOSE und die Dateinummer an. In unserem Beispiel geben wir ein:

```
CLOSE 21
```

Während Sie sich beim OPEN an manches erinnern müssen, ist es beim CLOSE umso weniger.

Das nächste Kommando bewirkt das Schreiben von Daten auf Band. Wir führen dies mit dem PRINT#-Kommando aus. Das Format ist

```
PRINT#,N,D
```

wobei "N" für die Dateinummer und "D" für die Daten steht. Für unser Beispiel würden wir eingeben:

```
PRINT#21, etc.
```

Wenn die Daten durch einen String dargestellt werden geben wir

```
PRINT#21,A$
```

ein und bei numerischen Daten:

```
PRINT#21,A (oder A% für Ganzzahlen).
```

Denken Sie daran, daß PRINT# nicht dasselbe ist wie PRINT. Sie können deshalb auch kein Fragezeichen (?) dafür einsetzen, wie es bei PRINT möglich ist. Wenn Sie ?# eingeben und das Programm laufen lassen, erhalten Sie eine Fehlermeldung. Wenn Sie das Programm auflisten, erscheint es als PRINT#. Um Ihnen dies zu zeigen, probieren Sie folgendes aus:

```
10 ?#1,5
20 PRINT#1,5
```

Geben Sie nun RUN ein, erhalten Sie ?SYNTAX ERROR IN 10. Geben Sie stattdessen RUN 20 ein, sehen Sie auf dem Bildschirm ?FILE NOT OPEN ERROR IN 20. Das Format in Zeile 20 ist zwar korrekt,

aber weil die Datei nicht eröffnet wurde, erhalten Sie eine Fehlermeldung. Listen Sie das Programm jetzt auf. Sie sehen, daß Zeile 10 und Zeile 20 nun identisch sind. Daher ist LIST nicht immer bei der Fehlersuche hilfreich. Denken Sie also daran und benutzen Sie bei PRINT# kein ? als Kürzel.

Genauso wie PRINT# Daten auf das Band ausgibt, holt INPUT# Daten vom Band in Ihren Computer. Es hat das gleiche Format wie PRINT# und benutzt die geöffneten Dateien, um numerische Variablen oder Strings einzulesen.

```
INPUT#21,A (oder A% für Ganzzahlen)   -  Zahlen
INPUT#21,A$                           -  Strings
```

Schließlich haben wir noch die GET#-Anweisung, die das gleiche Format wie INPUT# hat, jedoch wie die GET-Anweisung nur ein Zeichen nach dem anderen annimmt. Sie liest jedoch Kommas, Doppelpunkte und andere Zeichen, was INPUT# nicht kann. GET# wird nicht häufig benutzt, da die meisten Anwendungen das Einlesen längerer Strings erfordern. Möchten Sie jedoch spezielle Zeichen einlesen, was mit INPUT# nicht möglich ist, ist GET# sehr nützlich.

Wir haben also Kommandos zur Verfügung, um eine Datei zu eröffnen, auf eine Datei zu schreiben oder von ihr zu lesen, und um eine Datei zu schließen. Bevor wir weitermachen, müssen wir uns noch eine spezielle Variable, ST, ansehen. Die Variable ST ist reserviert, um zu prüfen, ob das Einlesen von Daten vom Band beendet ist. Wenn ST = 0 ist, sind noch mehr Daten zu erwarten. Die End-Of-File oder End-Of-Tape-Marke wurde noch nicht gelesen. Somit ist es möglich, nach einer IF/THEN-Abfrage mit der Variablen ST in einer Schleife zum Lesen zurückzuspringen. Es kann z.B. folgendes Format benutzt werden:

```
20 INPUT#21,A$
30 PRINT A$
40 IF ST = 0 THEN 20
```

Zeile 40 prüft ab, ob sich noch Daten auf dem Band befinden. Wenn ja, verzweigt das Programm zu Zeile 20, um diese einzulesen.

Wir haben jetzt alle Kommandos zum Lesen und Schreiben von Dateien auf Band kennengelernt. Sehen wir uns noch eine praktische Anwendung dazu an. Wir erstellen eine Liste mit den Telefonnummern unserer Freunde. Zuerst müssen wir eine Liste zur Eingabe der Da-

ten erstellen und diese dann auf Band sichern. Dann schreiben wir ein Programm, um die Namen und Telefonnummern einzulesen.

```
10 PRINT CHR$(147)
20 REM *** DATENEINGABE ***
30 INPUT "WIE VIELE NAMEN WOLLEN SIE EINGEBEN";N%
40 PRINT.DIM NA$(N%),PH$(N%)
50 FOR I=1 TO N%
60 PRINT "NAME #";I;:INPUT "VORNAME NACHNAME";NA$(I)
70 INPUT "TELEFONNUMMER (XXXX/XXXXXXX)";PH$(I)
80 NEXT I
100 REM *** DATEN AUF BAND SPEICHERN ***
110 OPEN1,1,1"TELEFON"
120 FOR I=1 TO N%
130 PRINT#1,NA$(I)
140 PRINT#1,PH$(I)
150 NEXT I
160 CLOSE1
```

Legen Sie ein leeres Band ein und spulen Sie die Kassette zurück. Geben Sie RUN und die gewünschten Daten ein. Sobald Sie die Play- und Recordtaste Ihres Kassettengerätes drücken, leert sich der Bildschirm und das Band dreht sich. Sind alle Informationen abgespeichert, stoppt das Band und auf dem Bildschirm erscheint die Meldung, daß alle Ihre Daten gespeichert wurden. (Das Speichern auf Band dauert im Vergleich zur Diskette relativ lange. Es ist zu Beginn ratsam, zu Testzwecken nur wenige Namen einzugeben.)

Lassen Sie uns kontrollieren, ob alles nach Plan gelaufen ist. Dazu brauchen wir ein Programm, das unsere Daten einliest. Wir benutzen dazu INPUT#. Da sowohl die Namen als auch die Telefonnummern als String abgespeichert wurden, lesen wir beides mit einer einzigen Stringvariablen ein. Wir nennen sie DA$ für "DATEN STRING". (Denken Sie daran, Ihr Band zurückzuspulen, bevor Sie RUN eingeben.)

```
10 PRINT CHR$(147)
20 OPEN1,1,0,"TELEFON"
30 INPUT#1,DA$
40 PRINT DA$
50 IF ST=0 THEN GOTO 30
60 CLOSE1
```

Wenn Sie RUN eingeben, werden Sie mit PRESS PLAY ON TAPE aufgefordert, Ihren Kassettenrecorder in Betrieb zu nehmen. Der Bildschirm leert sich erneut und nach einer Weile erscheint Ihre Textausgabe mit allen Namen und Telefonnummern, die Sie eingegeben haben. Hier werden Sie sagen: "Moment mal. Ich habe die Daten durch zwei verschiedene Stringelemente eingegeben, und dieses Programm liest nur eine Stringvariable! Was ist mit den Elementen geschehen, und wie ist es möglich, alle Informationen mit einer Stringvariable zurückzuholen?"

Die Antwort auf diese Frage beruht auf der Art, wie die Daten eingegeben werden. Nach der Eröffnung der Datei haben wir ihr die Daten mit INPUT# entnommen. Sobald sie im Hauptspeicher waren, haben wir sie mit unserem BASIC-PRINT-Befehl ausgegeben und nicht mit der PRINT#-Anweisung, mit der wir die Daten auf das Band ausgegeben haben. Der Computer kümmert sich nicht darum, ob die bereitgestellten Daten Namen oder Telefonnummern sind. Sie sind nur ein String, den er ausgibt, sobald sich dieser im Speicher befindet. Da der Bildschirm zu diesem Zeitpunkt noch dunkel war, konnten wir dies nicht verfolgen. In Zeile 50 prüft das Programm, ob sich noch mehr Informationen in der Datei befinden, nimmt diese auf und gibt den nächsten String aus, ohne Rücksicht darauf, ob es sich dabei um einen Namen oder eine Telefonnummer handelt. Um dies zu testen, geben Sie nur PRINT DA$ im Kommandomodus ein, und der letzte Satz erscheint am Bildschirm.

Bauen wir unserer Programm noch ein bißchen aus. Wenn Sie es benutzen, um die Telefonnummern Ihrer Freunde einzugeben, wird die Liste wahrscheinlich länger als ein Bildschirm werden. Sie können deshalb nur die Namen und Telefonnummern lesen, die sich noch auf dem Bildschirm befinden. Wir brauchen also ein Programm, das nach einem speziellen Namen sucht, diesen findet, die Datei schließt und den gefundenen Namen und die Telefonnummer auf dem Bildschirm ausgibt.

```
10 PRINT CHR$(147)
20 INPUT "WELCHEN NAMEN WOLLEN SIE SUCHEN";NA$
30 OPEN1,1,0,"TELEFON"
40 INPUT#1,DA$
50 IF DA$=NA$ THEN GOTO 100
60 IF ST=0 THEN GOTO 40
70 PRINT "NAME NICHT GEFUNDEN"
80 CLOSE1
90 END
```

```
100 REM *** NAME UND TELEFONNUMMER EINGEBEN ***
110 PRINT DA$:REM NAMEN DRUCKEN
120 INPUT#1,DA$:REM TELEFONNUMMER EINLESEN
130 PRINT DA$.REM TELEFONNUMMER DRUCKEN
140 CLOSE1
```

Wir haben nun ein nettes Programm für die Speicherung einer Telefonliste und das Auffinden eines einzelnen Namens mit Telefonnummer auf Band. Das nächste Problem ist: wie aktualisiere ich die Datei, ohne alle Daten erneut einzugeben? Dies trifft z.B. zu, wenn Sie in eine bestehende Telefonliste einen neuen Namen einfügen möchten. Können wir das? Natürlich! Aber zuerst müssen wir alle Namen vom Band in den Speicher lesen und sie dann wieder zurückzuschreiben. Dafür gibt es verschiedene Möglichkeiten. Unser Beispiel ist nur eine davon. Wir machen folgendes:

1. Laden Sie alle Namen und Telefonnummern in eine Tabelle.
2. Geben Sie neue Namen und Telefonnummern am Tabellenende ein.
3. Spulen Sie das Band zurück und speichern Sie die Daten erneut ab.

```
10 PRINT CHR$(147)
20 DIM NA$(30),PH$(30):REM PLATZ FÜR DIE NAMEN + RESERVE
30 OPEN1,1,0,"TELEFON"
40 N=0:REM ZÄHLER INITIALISIEREN
50 INPUT#1,NA$(N)
60 INPUT#1,PH$(N)
70 N = N + 1
80 IF ST=0 THEN 50
90 CLOSE1
100 REM *** DATEN ANFÜGEN ***
110 INPUT "WIE VIELE NEUE NAMEN ANFÜGEN";NN
120 FOR I=(N+1) TO (N+NN)
130 INPUT "NAME";NA$(I)
140 INPUT "TELEFON";PH$(I)
150 NEXT I
200 REM *** ALTE UND NEUE DATEN SPEICHERN ***
210 NP = N + NN
220 OPEN1,1,1,"TELEFON"
230 FOR I = 0 TO NP
240 PRINT#1,NA$(I)
250 PRINT#1,PH$(I)
260 NEXT I
270 CLOSE1
```

Vergewissern Sie sich, daß Ihr Band zurückgespult ist, sobald alle Daten im Speicher sind. Fügen Sie zur Sicherheit folgende Zeile an:

```
105 PRINT CHR$(147):PRINT "SPULEN SIE IHR BAND ZURÜCK!":PRINT:
INPUT "WEITER MIT <RETURN>";RT$
```

Jetzt können Sie es nicht vergessen.

Sehen wir uns nun an, wie Sie Sprites auf Band sichern können. Weiterhin werden wir sehen, wie wir die Informationen vom Band laden und ein Programm starten können, das diese Daten benutzt. An dieser Stelle jedoch eine Warnung: manchmal sind auf dem Band mehr Informationen, als Sie brauchen. Es ist daher wichtig, in den Speicher nur das zu laden, was Sie brauchen. Alles andere wird ignoriert. Dies ist ein bißchen unbequem, da Sie die gesamten Daten im Auge behalten müssen und auch die Abgrenzungen Ihres Sprites kennen müssen. Da diese Informationen jedoch in jedem Fall notwendig sind, dürfte es Ihnen nicht allzu schwer fallen.

Zuerst erstellen wir ein Sprite und sichern es auf Band. Es soll wie eine Rakete aussehen.

```
10 PRINT CHR$(147):REM***ERSTELLUNG DES SPRITES***
20 DIM RS(63)
30 FOR I = 0 TO 32:READ RS(I):NEXT
40 DATA 30,0,0,31,0,0,31,128,0,31,255,192,192,31,255,240,31,
255,248
50 DATA 31,255,240,31,255,192,31,128,0,31,0,0,31,0,0
60 FOR I = 33 TO 62:RS(I)=0:NEXT I:REM REST MIT NULLEN AUFF.
100 REM *** BEGINN DATENSPEICHERUNG AUF BAND ***
110 OPEN1,1,1,"RAKETE"
120 FOR I = 0 TO 62
130 PRINT#1,RS(I)
140 NEXT I
150 CLOSE1
```

Wir haben das Sprite in der uns gewohnten Weise erstellt. Sorgen Sie dafür, daß kein "Schund" in das Sprite gerät, indem Sie die leeren Reihen des Sprite mit "0" füllen. Wenn Sie jetzt die Daten vom Band laden, wissen Sie, egal wie das Sprite aussieht, daß 63 (0 bis 62) "Informationsteile" geladen werden. Daher müssen wir beim Laden der Daten nicht lang überlegen, wie diese im Speicher stehen. Durch die Benutzung der maximalen Anzahl an Bytes benöti-

gen die kleinsten wie die größten Sprites jeweils 63 Bytes. Wenn das Sprite also kleiner als 63 Bytes ist, füllen Sie alle leeren Bytes mit "0" auf.

Laden Sie nun das Sprite vom Band und lassen Sie es in einem Programm ablaufen. Speichern Sie in jedem Fall das folgende Programm ab, mit dem Sie jedes Sprite vom Band lesen und dann ablaufen lassen können, denn dies spart eine Menge Zeit. Sie laden nur das Programm, bringen das Band an den Beginn des Sprites und lassen das Programm dann ablaufen. Sie können dies mit jedem gewünschten Sprite durchführen.

```
10 PRINT CHR$(147)
20 BG = 53248:TU = BG + 21:S1 = 2:L  2040:SS = 832:SL = 13
30 X1 = BG + 2:Y1 = BG + 3
40 REM *** ENDE DES DEFINITIONSBLOCKS ***
50 REM
100 REM *** DATEN VOM BAND EINLESEN ***
110 OPEN1,1,0,"RAKETE":REM SPRITE EINLESEN
120 DIM R(64)
130 FOR I = 0 TO 62
140 INPUT#1,R(I)
150 NEXT I
160 CLOSE1:REM ALLE DATEN EINGELESEN, DATEI SCHLIESSEN
200 REM *** SPRITE ERSTELLEN ***
210 FOR J = 0 TO 62
220 POKE SS+J,R(J)
230 NEXT J
240 POKE TU,S1
250 POKE L+1,SL
260 POKE Y1,100
270 FOR H = 0 TO 255
280 POKE X1,H:NEXT H
290 GOTO 270
```

Ohne Zweifel sehen Sie hier unser Sprite-Programm von Kapitel 7. Sie bekommen Ihre Spritewerte jedoch nicht von den DATA-Zeilen, sondern vom Band. Beachten Sie, daß wir nicht die Variable ST zur Überprüfung benutzen, ob alle Werte eingelesen wurden. Stattdessen schließen wir die Datei, sobald alle 63 Spritewerte geladen sind. Dies schließt die Gefahr aus, daß sich "fremde" Daten vom Band einschleichen. Wir sollten auch die Möglichkeit auf diese Weise mehrere Sprites zu laden, nicht außer acht lassen. Sie können dazu entweder mehrere Sprites aus der gleichen Datei, oder,

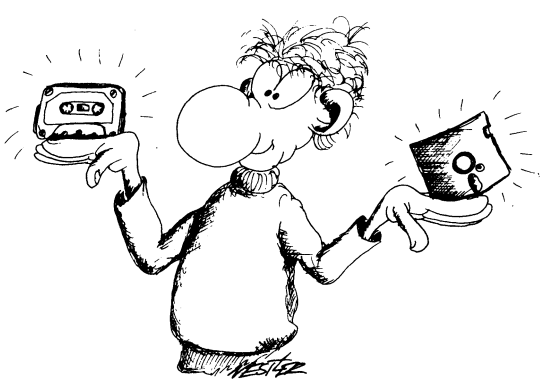
was besser ist, aus verschiedenen Dateien einlesen. Denken Sie dabei nur einfach an die Methoden zur Erzeugung und Behandlung von Sprites, die wir in Kapitel 7 diskutiert haben, mit dem kleinen Unterschied, daß wir die Daten vom Band und nicht aus den DATA-Zeilen lesen.

8.3 Sequentielle Daten auf Diskette

Wenn Sie kein Diskettenlaufwerk haben, können Sie diesen Abschnitt überspringen und zum nächsten Kapitel übergehen. Andernfalls ist das folgende von großem Interesse. In gewisser Hinsicht ist das Speichern von Daten auf Diskette ähnlich wie das Speichern auf Band; allerdings geht der Speicher- und Ladeprozeß sehr viel schneller vonstatten. Natürlich können alle unsere Beispiele aus dem vorangegangenen Abschnitt ebenso mit dem Diskettensystem durchgeführt werden. Es sind nur winzige Änderungen des Formats notwendig. Zu Beginn sehen wir uns an, wie wir mit kleinen Änderungen des Formats Daten auf Diskette speichern können. Wir werden dabei das OPEN-, CLOSE-, INPUT#-, PRINT#- und GET#-Kommandos vorstellen.

OPEN. Um einen Diskettenkanal zu öffnen, geben wir das Gerät mit der Ziffer "8", anstatt "1" (beim Band), an. Nachdem die Diskette mit der Filenummer "15" initialisiert wurde, öffnen wir eine weitere Datei mit der Nummer "9". Dazu geben wir ein:

```
OPEN 9,8, etc.
```



Anstatt die Codenummern 0, 1 und 2 zum Schreiben oder Lesen und Schreiben der End-Of-Tape-Marke zu benutzen, verwenden wir READ (lesen) und WRITE (schreiben), oder kurz, R und W. Wir stellen dies jedoch unterschiedlich dar. Wir eröffnen eine Datei, indem wir zuerst die Dateinummer und die Gerätenummer eingeben, und dann erst eine zweite Adresse und die restlichen Filehandling-Kommandos in Anführungszeichen. Die zweite Adresse sollte nicht 15 sein, da diese bei der Fehlerbehandlung benutzt wird. Nehmen Sie auch nicht die Zahlen 0 und 1, da diese zum Speichern und Laden von Programmen durch das Filehandlingsystem gebraucht werden. Die Nummern der zweiten Adresse können zwischen 2 und 14 liegen. Wir benutzen hier die "9".

```
OPEN 9,8,9, ETC.
```

Wir beginnen die Angaben in Anführungszeichen mit der Laufwerknummmer "0" bei einem Laufwerk, gefolgt von einem Doppelpunkt und dem Dateinamen.

```
OPEN 9,8,9,"0:TELEFONLISTE, etc."
```

Als nächstes geben wir den Dateityp an. Im Moment benutzen wir sequentielle (fortlaufende) Dateien, angedeutet durch SEQ. Zuletzt geben wir READ oder WRITE an, je nachdem, ob wir Informationen auf Diskette speichern (WRITE) oder diese vom Band einlesen (READ) möchten. Die gesamte Routine zum Eröffnen von Dateien sieht nun so aus:

```
OPEN 9,8,9,"0:TELEFONLISTE,SEQ,WRITE"
```

Glücklicherweise benutzen die INPUT#-, PRINT#- und GET#-Kommandos das gleiche Format wie beim Band. Die Zahl, die dem Kommando folgt, ist die zweite Adresse. Wenn wir also in unserem Beispiel mit PRINT# ausgeben möchten, schreiben wir

```
PRINT#9,
```

da "9" unsere zweite Adresse ist. (Beachten Sie, daß die erste Adresse ebenfalls die "9" ist. Durch Benutzen der gleichen Zahl für die erste und zweite Adresse verringern wir die Gefahr der Verwechslung.)

Um zu sehen, wie all dies funktioniert, führen wir unser Telefonprogramm, das wir als Bandbeispiel erstellt haben, nochmals auf.

Der Dateneingabeblock ist identisch. Wir bearbeiten also nur den Block, der die Daten auf Diskette ausgibt.

```
100 REM *** SICHERN VON DATEN AUF DISKETTE ***
110 OPEN 9,8,9,"0:TELEFONLISTE,SEQ,WRITE"
120 FOR I = 1 TO N%
130 PRINT#9,NA(I)
140 PRINT#9,PH(I)
150 NEXT I
160 CLOSE9
```

Wie Sie sehen, liegt der Hauptunterschied zwischen Band und Diskette am Format in Zeile 110. Das Format zum Ausgeben von Daten ist bei Band und Diskette identisch. Das gleiche gilt fast uneingeschränkt auch für den Abruf der Daten. Beachten Sie, daß Sie die Variable ST auch bei Disketten I/O (Ein- und Ausgabe) verwenden können.

```
10 PRINT CHR$(147)
20 OPEN 9,8,9,"0:TELEFONLISTE,SEQ,READ"
30 INPUT#9,DA$
40 PRINT DA$
50 IF ST 0 = THEN GOTO 30
60 CLOSE9
```

Um Zeit zu sparen, können Sie auch "S" für "SEQ" und "R" für "READ" schreiben. Mit diesen Abkürzungen sieht Zeile 20 folgendermaßen aus:

```
20 OPEN 9,8,9,"0:TELEFONLSTE,S,R"
```

Genauso können wir dies auch mit den anderen Dateistatusworten durchführen. Sie können diese Abkürzungen jedoch nicht für Dateinamen verwenden.

Bevor wir uns noch mehr Methoden zur Benutzung des Diskettensystems ansehen, möchte ich Ihnen noch die Technik zum Ändern von Dateien zeigen. Wie Sie sich erinnern, mußten Sie beim Band erst alle Daten der alten Datei einlesen, neue Daten hinzufügen, das Band zurückspulen und somit die alte Datei mit der neuen überschreiben. Die Diskette arbeitet auf die gleiche Weise. Aber anstatt sie zurückzusetzen, benutzen wir beim Ändern von Daten ein spezielles Format des OPEN-Befehls. Der folgende Block zeigt Ihnen die Vorgehensweise.

```
200 REM ***SCHREIBEN VON ALTEN UND NEUEN DATEN AUF DISKETTE
210 NP = N + NN:REM KOMBINIERTE SUMME ALLER NAMEN
220 OPEN 9,8,9,"§0:TELEFONLISTE,S,W"
230 FOR I = 0 TO NP
240 PRINT#9,NA$(I)
250 PRINT#9,PH(I)
260 NEXT I
270 CLOSE9
```

Der Schlüssel zum Überschreiben einer Datei ist das Symbol "§". Wir benutzen dieses Format auch, um bestehende Dateien zu überschreiben.

Da wir inzwischen eine Anzahl individueller Programme erstellt haben, schreiben wir jetzt ein Programm, das 1) Dateien erstellt, 2) eine einzelne Datei liest und diese 3) an eine bestehende Datei anfügt. Anstelle von Namen und Telephonnummern nehmen wir jetzt Namen und Adressen.

```
10 PRINT CHR$(147):RESTORE:CLR
20 PRINT "***";CHR$(18);"DATEIVERWALTUNG";CHR$(146);"***"
30 PRINT:FOR I = 1 TO 5:PRINT I"-":PRINT:NEXT
40 PRINT CHR$(19):PRINT:PRINT:FOR I = 1 TO 5:READ D$:PRINT
SPC(5);D$:PRINT:NEXT
50 DATA NEUE DATEI ERSTELLEN, SÄTZE HINZUFÜGEN, ALLE DATEIEN
LESEN, EINE DATEI LESEN
60 DATA EXIT
70 PRINT:PRINT CHR$(18);"KENNZIFFER FÜR VERARBEITUNG";CHR$
(146)
80 GET A:IF A < 1 THEN 80
90 ON A GOTO 100,200,400,500,700
100 REM *** DATEI ERSTELLEN ***
110 PRINT CHR$(147):PRINT:PRINT
120 INPUT "WIE VIELE NAMEN";N%
130 OPEN9,8,9"0:ADRESSEN,S,W"
140 PRINT#9,N%:REM ANZAHL ADRESSEN WIRD IN DATEI GESCHRIEBEN
150 DIM NA$(N%),ST$(N%),PL$(N%),WO$(N%)
160 FOR I = 1 TO N%:GOSUB 1000:GOSUB 2000:NEXT
170 CLOSE9
180 GOTO 10
200 REM *** SÄTZE HINZUFÜGEN ***
210 PRINT CHR$(147):PRINT:PRINT
220 INPUT "ANZAHL ANZUFÜGENDER SÄTZE";NN%
230 OPEN9,8,9,"0:ADRESSEN,S,R"
```

```
240 INPUT#9,N%:REM ANZAHL EXISTIERENDER SÄTZE ERMITTELN
250 NP% = N% + NN%
260 DIM NA$(PN%),ST$(NP%),PL$(NP%),WO$(NP%)
270 FOR I = 1 TO N%.GOSUB 3000:NEXT
280 CLOSE9
300 PRINT#9,NP%
310 FOR I = (N%+1) TO NP%:GOSUB 3000:NEXT
330 CLOSE9
320 FOR I=1 TO NP%:GOSUB 2000:NEXT
330 CLOSE9
340 GOTO 10
400 REM *** UNTERPROGRAMM LESEN ***
410 OPEN9,8,9,"ADRESSEN,S,R"
420 INPUT#9,N%
430 DIM NA$(N%),ST$(N%),PL$(N%),WO$(N%)
440 FOR I = 1 TO N%:GOSUB 3000:NEXT
450 CLOSE9
460 FOR I = 1 TO N%:GOSUB 4000:NEXT
470 PRINT:PRINT "WEITER MIT BELIEBIGER EINGABE"
480 GET A$:IF A$ = "" THEN 480
490 GOTO 10
500 REM *** SUCHROUTINE ***
510 PRINT CHR$(147)
520 INPUT "WELCHEN NAMEN SUCHEN";NA$
530 OPEN9,8,9,"ADRESSEN,S,R"
540 INPUT#9,N%
550 DIM NA$(N%),ST$(N%),PL$(N%),WO$(N%)
560 FOR I = 1 TO N%:GOSUB 3000
570 IF NA$(I) = NA$ THEN GOSUB 4000
580 NEXT
590 CLOSE9
600 PRINT:PRINT:PRINT:PRINT "WEITER MIT BELIEBIGER EINGABE"
610 GET AN$:IF AN$ = "" THEN 610
620 GOTO 10
700 REM *** ENDE ***
710 END
1000 REM *** UNTERPROGRAMM EINGABE ***
1010 PRINT "NAME #";I
1020 INPUT INPUT NA$(I)
1030 INPUT "STRASSE";ST$(I)
1040 INPUT "PLZ";PL$(I)
1050 INPUT "ORT";WO$(I)
1060 RETURN
2000 REM *** UNTERPROGRAMM PRINT# ***
```

```
2010 PRINT#9,NA$(I)
2020 PRINT#9,ST$(I)
2030 PRINT#9,PL$(I)
2040 PRINT#9,WO$(I)
2050 RETURN
3000 REM *** UNTERPROGRAMM INPUT# ***
3010 INPUT#9,NA$(I)
3020 INPUT#9,ST$(I)
3030 INPUT#9,PL$(I)
3040 INPUT#9,WO$(I)
3050 RETURN
4000 REM *** UNTERPROGRAMM DRUCK ***
4010 PRINT NA$(I)
4020 PRINT S4$(I)
4030 PRINT
4040 PRINT PL$(I)+" "+WO$(I)
4050 PRINT:PRINT
4060 RETURN
```

Das war ein langes Programm. Wenn Sie Programme von dieser Länge erstellen, ist es besser, wenn Sie Ihr Programm nach jeweils 10 - 15 Zeilen abspeichern. Sie können so das Programm wieder holen, wenn etwas schief ging. Es ist wichtig, verschiedene Aspekte des Programms zu beachten, so wie z.B. das neue Kommando CLR. Das CLR-Kommando löscht alle Variablen und Tabellen. Es ist für jene Programme, mit denen Sie verschiedenes ausführen möchten, während sich das Programm noch im Speicher befindet. Sie wollen z.B. Daten an Ihre Adressenliste anfügen und anschließend einen einzelnen Namen lokalisieren. Durch Löschen der Variablen und Tabellen verhindern Sie, daß Sie falsche Werte erhalten, wenn Sie zum Menü zurückkommen.

Ein weiterer wichtiger Punkt, den Sie beachten sollten, ist die Verwendung von Unterroutinen. Diese machen nicht nur das Programm besser lesbar; Sie sparen sich durch Unterroutinen auch Programmierzeit. Zum Beispiel wird sowohl in der "READ"-, als auch in der "SUCH"-Routine die "INPUT#"-Unterroutine verwendet. Anstatt diese Programmzeilen mehrfach einzugeben, springt das Programm zur Unterroutine. Im nächsten Kapitel stelle ich noch eine Unterroutine vor, die zum Ausgeben aller Namen und Adressen auf den Drucker dient. So schreiben Sie nicht das gesamte Programm neu, sondern fügen nur die notwendige Unterroutine hinzu.

8.4 Zusammenfassung

In diesem Kapitel haben Sie gelernt, durch Abspeichern von Dateien auf Band oder Diskette Zeit zu sparen. Sie können auf Band Daten abspeichern, die Sie später mit einem Programm verarbeiten. Dies ist sehr nützlich, da es Ihnen ermöglicht, heute Daten einzugeben, die Sie erst später verarbeiten, ohne sie erneut eingeben zu müssen. Das kann natürlich mit einem einzigen Programm, wenn Sie dabei PRINT- und DATA-Zeilen verwenden, ebenso gemacht werden. Der Benutzer ist jedoch an das Programm gebunden, das die betreffenden DATA-Zeilen enthält. Nach dem Abspeichern auf Band oder Diskette können Sie die Daten von verschiedenen Programmen aus verarbeiten. Das ist bei Sprites von besonderem Nutzen.

Wenn Sie ein Diskettensystem benutzen, ist es möglich, Daten, wie auf dem Band, in sequentieller Folge abzuspeichern. Die Diskette erlaubt jedoch einen viel schnelleren Zugriff auf die Daten und es ist möglich, mit Datendateien auf Diskette, verschiedene Dinge mit einem einzigen Programm durchzuführen. Das "File Master"-Programm zeigt, wie ein einziges Programm dazu benutzt wird, um einzelne oder mehrere Dateien zu erstellen, sie zu lesen oder Daten zu ergänzen. Es muß zwar alles mit großer Vorsicht durchgeführt werden, Sie vergrößern aber auch die Leistungsfähigkeit Ihres Computers, wenn Sie mit sequentiellen Dateien arbeiten. Die praktische Anwendung dieser Programme ist von großer Bedeutung.

9

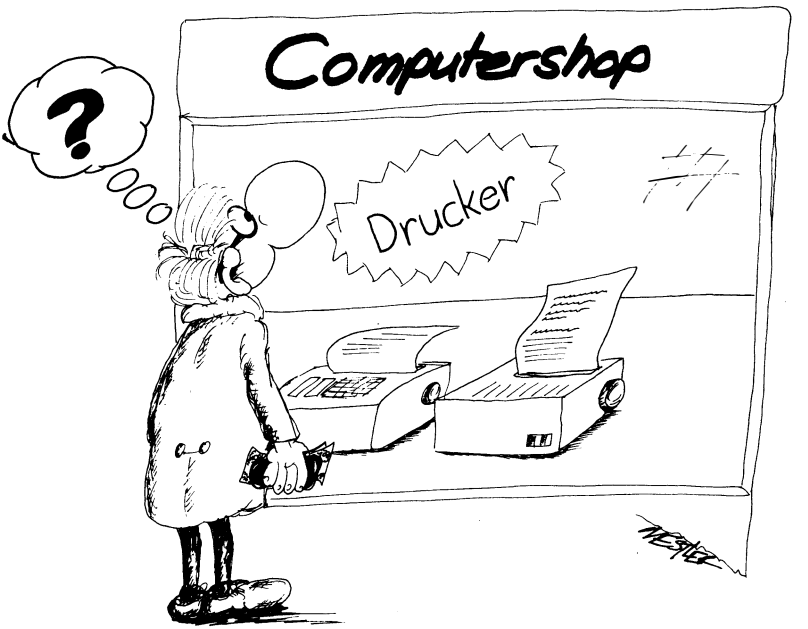
Ihr Drucker

Einführung

Sie sollten inzwischen so geübt sein, daß Sie Informationen auf den Bildschirm, auf Diskette oder Band ausgeben können. Mit PRINT "HALLO" geben Sie auf den Bildschirm aus. Mit SAVE oder PRINT# erzeugen Sie eine Ausgabe auf Band oder Diskette. In der gleichen Weise, wie Sie mit Bildschirm, Band oder Diskette in Verbindung treten, können Sie den Drucker "ansprechen". Er stellt nur ein weiteres "Ausgabeziel" dar. Sie können jedoch nicht mit LOAD, INPUT oder ähnlichen Befehlen vom Drucker Daten einlesen.

Um Informationen auf den Drucker auszugeben und die speziellen Möglichkeiten des Druckers zu nutzen, müssen Sie bestimmte Prozeduren lernen, die wir bis jetzt noch nicht besprochen haben. Während Ihnen einiges aus diesem Kapitel bereits bekannt ist, zum Beispiel was die Kommandos betrifft, stellt doch die Anordnung dieser Kommandos etwas Neues dar. Wir sehen uns auch an, wie wir den Drucker für Aufgaben verwenden können, die mit dem Bildschirm nur schlecht durchzuführen sind. Lange Programmlistings können Sie z.B. besser auf dem Drucker ausgeben, da sie am Bildschirm über den oberen Rand "ins Nichts" verschwinden. Wie in Kapitel 8 erstellen wir ein kleines Programm zur Speicherung von Telefonnummern und ein weiteres zur Speicherung von Adressen. Mit Hilfe eines Druckers können Sie die Telefonliste ausgeben oder Adressenaufkleber drucken.

Es gibt sehr viele Drucker auf dem Computermarkt. Um es so einfach wie möglich zu machen, nehmen wir für unsere Beispiele den VIC-1515-Drucker von Commodore. Dieser Drucker liefert alle Grafik- und Textmöglichkeiten, die Sie brauchen, und ist sehr einfach mit dem COMMODORE-64-System zu verbinden. Verglichen mit anderen, ist er auch sehr preisgünstig. Haben Sie einen anderen Drucker mit dem passenden Interface für Ihren COMMODORE-64, müssen Sie sich gut mit dem Druckerhandbuch vertraut machen. Leider sind manche Druckerhandbücher für Anfänger schwer zu lesen, da sie oft sehr technisch geschrieben sind. Achten Sie deshalb sehr genau auf die Codes, welche die speziellen Möglichkeiten Ihres Druckers ein- oder ausschalten. Dies wird gewöhnlich mit dem CHR$-Kommando von BASIC gemacht. Es ist also das wichtigste, daß Sie den Anweisungen dieses Buches folgen und gleichzeitig die geeigneten Codes Ihres Druckerhandbuches verwenden.



Bevor Sie einen Drucher kaufen

Der wichtigste Aspekt beim Kauf eines Druckers ist das passende Interface für Ihren COMMODORE-64. Sehr oft erzählen enthusiastische Verkäufer dem Anwender alle Qualitäten eines Druckers und nehmen einfach an, daß er mit jedem Computer zusammen funktioniert. Das ist jedoch nicht der Fall! Damit ein Drucker mit einem Computer arbeitet, braucht er das passende Interface. Vergewissern Sie sich deshalb vor dem Kauf eines Druckers, der nicht, wie z.B. der VIC-1515, speziell für den COMMODORE-64 hergestellt wurde, ob Sie auch das geeignete Interface dafür bekommen. Der sicherste Weg zu erfahren, ob der Drucker mit einem COMMODORE-64 zusammen arbeitet, ist eine Testdemo. Die VIC-1515 und VIC-1525 Drucker arbeiten problemlos mit dem COMMODORE-64.

9.1 Die Ausgabe von Text auf dem Drucker

Das erste, was Sie mit Ihrem Drucker machen werden, ist die Ausgabe von Text als "Hardcopy" ("Hardcopy" ist ein sehr eindrucksvoller Ausdruck der Computersprache: er heißt nichts anderes als die Druckausgabe auf Papier. Benutzen Sie ihn, um Ihre Freunde zu verblüffen!). Wie beim Kassettenrecorder oder der Diskettenstation ist es notwendig, vor der Informationsausgabe einige Schritte durchzuführen. Lassen Sie uns diese wiederholen.

OPEN - Zuerst schaffen Sie damit eine Verbindung zu Ihrem Drucker. Beim VIC-1515 befindet sich auf der Rückseite ein Schalter mit der Gerätenummer 4 oder 5. Wir benutzen in unseren Beispielen die "4". Sehen Sie also nach, ob sich der Schalter in Position "4" befindet. (Erinnern Sie sich, daß die Gerätenummer Ihrer Diskettenheit "8" ist.) Die Reihenfolge für das OPEN beim Drucker ist folgende: Sie geben zuerst eine Nummer zwischen 1 und 255 an (wir benutzen die Glückszahl 7) und anschließend die Gerätenummer 4. Hier ein Beispiel:

```
OPEN7,4
```

Jetzt ist Ihr Drucker bereit, Instruktionen für die logische Dateinummer "7" zu empfangen.

CMD - Das CMD-Kommando weist Ihren Computer an, Daten an Ihren Drucker zu senden. Sie verwenden dazu die Dateinummer (in unserem Beispiel die 7) und nicht die Gerätenummer (4). Sie geben z.B. folgendes ein:

```
CMD7
```

Ihr Drucker wird "angekurbelt" und druckt READY aus, wie Sie es vom Bildschirm gewohnt sind. Geben Sie folgendes ein:

```
FOR I = 1 TO 10:PRINT:NEXT
```

Das Papier im Drucker wird 10 Zeilen vorgeschoben; bisher löste dieser Befehl einen Zeilenvorschub auf dem Bildschirm aus. Die Ausgabe erscheint jetzt so lange auf dem Drucker, anstatt auf dem Bildschirm, bis Sie diese Verbindung wieder unterbrechen.

Es gibt zwischen der Ausgabe auf dem Drucker und dem Bildschirm einen wichtigen Unterschied. Wenn Sie

```
FOR I = 1 TO 80:PRINT "X";:NEXT
```

eingeben, erscheinen alle 80 "X" in einer Zeile. Bei Benutzung des gleichen Kommandos am Bildschirm, würde es in zwei Zeilen erscheinen, weil Ihr Drucker 80 Spalten, statt der 40 Spalten des Bildschirms ausgibt.

PRINT# - Sie erinnern sich, daß wir PRINT# im Programm dazu verwenden, Informationen auf Band oder Diskette auszugeben. Nun, der Drucker wendet das gleiche Prinzip an. Nehmen wir den Fall an, Sie möchten nur wenig Daten im Programm ausgeben, und nicht alle davon auf dem Drucker. Da Sie aber das CMD-Kommando eingegeben haben, geht alles, was Sie normal auf dem Bildschirm sehen, auf den Drucker.

Benutzen Sie jedoch PRINT#, wird nur diese Information zum Drucker geschickt. Angenommen, Sie möchten, daß Sie am Bildschirm mit "Name?" aufgefordert werden, und daß dieser nach Eingabe sofort am Drucker ausgegeben wird, dann benutzen Sie PRINT#. Das Format ist

```
PRINT#7,NA$
```

oder

```
PRINT#7,"CHARLIE BROWN"
```

oder

```
PRINT#7,12345
```

Lassen Sie uns ein kleines Programm ausprobieren, das Namen auf dem Drucker ausgibt. Das Programm spricht sowohl den Drucker als auch den Bildschirm an.

```
10 PRINT CHR$(147)
20 PRINT:PRINT:PRINT "DRUCKER EINSCHALTEN"
30 PRINT:PRINT:PRINT "WEITER MIT BELIEBIGER EINGABE"
```

```
40 GET A$:IF A$ = "" THEN 40
50 PRINT CHR$(147)
60 OPEN7,4
70 INPUT "ZU DRUCKENDER NAME";NA$
80 PRINT#7,NA$
90 INPUT "WEITERE NAMEN (Y/N)";AN$
100 IF AN$ = "Y" THEN 70
110 CLOSE7
120 END
```

CLOSE - Das letzte Kommando, mit dem Sie Ihren Drucker ansprechen, ist CLOSE. Wie wir im Programm oben sehen, schließt das Kommando den Kanal zum Drucker und schaltet ihn ab. Größtenteils arbeitet das CLOSE hier genau so wie bei der Diskette oder beim Band; es kommt jedoch noch etwas hinzu. Bevor Sie den Kanal zum Drucker schließen, müssen Sie erst PRINT#<fn> eingeben. Wenn Sie deshalb mit OPEN einen Kanal zum Drucker öffnen und das CMD-Kommando benutzen, müssen Sie vor dem CLOSE-Kommando zuerst PRINT# angeben.

Zum Beispiel:

```
OPEN7,4
CMD 7
PRINT "HALLO HALLO"
PRINT#7
CLOSE7
```

Wenn Sie das CLOSE-Kommando nicht in Verbindung mit dem PRINT# Kommando aussenden, treten Probleme auf.

Auflisten von Programmen

Da das Auflisten von Programmen über den Drucker sehr geeignet für die Dokumentation und die Fehlersuche ist, wäre es gut ein Hilfsprogramm dafür zu haben. Daher schreiben wir jetzt ein Programm, das Ihr Programm auf den Drucker ausgibt. Wir werden es kurz halten, so daß wir es zu Beginn jedes Programms anhängen können. Zuerst laden Sie ein Programm in den Speicher und fügen Sie folgende Zeilen hinzu:

```
1 OPEN7,4
2 CMD7
3 LIST 5-
4 END
```

Wenn Sie RUN eingeben, schaltet der Drucker auf "bereit" und listet das Programm von Zeilennummer 5 bis ans Ende auf. Wenn Sie fertig sind, geben Sie noch PRINT#7 und CLOSE7 ein, um die Datei und den Kanal zu schließen. Leider können Sie das CMD-Kommando vom Programm aus nicht umschalten, indem Sie PRINT# und CLOSE angeben, so wie dies bei Verwendung des PRINT#-Kommandos möglich war. Vergewissern Sie sich also, daß Sie den Drucker nach dem Listing ausgeschaltet haben, um ihn aus dem Kommandomodus zurückzuholen.

9.2 CHR$-Funktionen

Das Geheimnis bei der Benutzung von Druckern liegt im Verständnis der Control-Codes. Sie müssen wissen, was sie bedeuten und wie man sie anwendet. Folgende Aufstellung zeigt zum Beispiel eine Teilliste von Codes, mit denen ein CENTRONICS 737 arbeitet:

MNEMONIC	DEZIMAL	OKTAL	HEX	FUNKTION
ESC,SO	27,14	033,016	1B,0E	Breitschrift
ESC,DC4	27,20	033,034	1B,13	16.7 Zeichen/inch
ESC,DC1	27,17	033,021	1B,11	Proportionalschrift

Für Anfänger in der "Welt der Computer" könnte das von einem Besucher eines fremden Planeten erstellt sein. Es stecken jedoch wichtige Informationen in diesen Codes, und wenn Sie erst einmal wissen, wie Sie diese einsetzen können, ist es relativ einfach.

Beachten Sie zu Beginn nur die Spalten DEZIMAL und FUNKTION. In der ersten Spalte haben wir den dezimalen Code 27 und 14, um die Breitschrift einzuschalten. Um dem Drucker mitzuteilen, daß Sie die Breitschrift wünschen, benutzen Sie CHR$(27) + CHR$(14). Um dies Ihrem Drucker mitzuteilen, müssen Sie folgendes machen:

```
1.   OPEN7,4

2.   PRINT#7,CHR$(27) + CHR$(14) + "NACHRICHT"
```

Wenn Sie einen Centronics 737 oder 739 haben, würde dieser jetzt den String "NACHRICHT" in Breitschrift ausgeben. Ebenso müßten Sie bei der verdichteten Druckausgabe (16.7 cpi = characters per inch) CHR$(27) + CHR$(20) und bei der Proportionalschrift CHR$(27) + CHR$(17) verwenden. Wenn Sie den Dezimalcode haben, geben Sie diesen nur dem Drucker mit und Sie erreichen alle Ausgabeformen, von der Umstellung des Schriftbildes bis zum Aufruf der "Rückschritt-Funktion".

Bei anderen Druckern ist es das gleiche. Lassen Sie uns zum VIC-1515-Drucker zurückkehren. Wie wir sehen, benutzt der VIC-1515 oder VIC-1525, genau wie der Centronics-Drucker (oder jeder andere), CHR$ Kommandos, um die verschiedenen Möglichkeiten des Druckers auszunutzen.

Lassen Sie uns die verschiedenen CHR$-Kommandos in Verbindung mit dem VIC-1515 ansehen:

CHR$	**FUNKTION**
10 & 13	Zeilenvorschub
8	Grafikmodus
14	doppelte Breite
15	zurücksetzen auf Standardschriftdichte
16	Startadresse für den Druckbeginn
27	Escape (wird in Verbindung mit anderen Codes benutzt)
145	Cursor up
17	Cursor down
18	Umkehr der Druckrichtung
146	zurück in die normale Druckrichtung

Um Ihnen zu zeigen, wie die CHR$-Funktionen arbeiten, schreiben wir ein einfaches Programm, welches Ihren Namen auf dem Drucker ausgibt. Da Sie bereits wissen, wie Sie normalen Text ausgeben, beginnen Sie mit der verbreiterten Textausgabe. Ein Blick in die Tabelle sagt uns, daß wir dies mit CHR$(14) erreichen. Also werden wir es in unser Programm einbauen. (Beachten Sie das Fehlen der Interpunktionszeichen nach dem Komma, das dem PRINT#7 folgt.)

```
10 PRINT CHR$(147)
20 OPEN7,4:REM KANAL 7 FÜR EXTERNE EINHEIT NR. 4 (DRUCKER)
30 INPUT "IHR NAME ";NA$
40 PRINT#7,CHR$(14) NA$:REM BREITSCHRIFT
50 PRINT#7,CHR$(15):REM ZURÜCK ZU NORMALSCHRIFT
60 CLOSE7
```

Geben Sie RUN ein. Drucken Sie einige Namen aus und beachten Sie dabei die Breitschrift.

Bis jetzt haben wir noch nicht viel mit Groß- und Kleinbuchstaben gearbeitet. Wenn Sie aber Text auf dem Drucker ausgeben, brauchen Sie oft die Groß-/Kleinschreibung. So möchten Sie z.B. Namen nicht als

DOKTOR MICHAEL SCHMITT,

sondern als

Doktor Michael Schmitt

ausgeben. Wir brauchen dazu ein spezielles CHR$-Kommando. Beim VIC-1515-Drucker ermöglicht der CURSOR DOWN-Modus, CHR$(17), die Darstellung von Klein- und Großbuchstaben. Um dies zu erreichen, ändern Sie Zeile 40 und 50 unseres Programms folgendermaßen:

```
40 PRINT#7,CHR$(17) NA$:REM KLEIN/GROSSCHREIBUNG
50 PRINT#7,CHR$(145):REM ZURÜCK ZU GROSSBUCHSTABEN
```

Drücken Sie jetzt gleichzeitig die Commodore-Taste und die SHIFT-TASTE, um Ihren Computer in den Groß-/Kleinbuchstabenmodus zu bringen, und geben Sie RUN ein. Wenn Sie jetzt die Großbuchstaben mit Hilfe der SHIFT-Taste eingeben, werden auf dem Drucker sowohl Kleinbuchstaben als auch Großbuchstaben ausgegeben.

Hätten Sie das vor der Programmänderung probiert, würden Sie bei Verwendung der SHIFT-Taste statt Großbuchstaben Grafikzeichen in der Druckausgabe sehen, auch wenn Sie den Computer bereits in den Groß-/Kleinbuchstabenmodus gebracht hätten. Probieren Sie es mit dem Originalprogramm aus, um es selbst zu sehen.

Ein weiterer Trick ist die Verwendung der Groß- und Kleinbuchstaben, zusammen mit der Breitschrift. Sie müssen das Programm so ändern, daß es beide CHR$-Kommandos enthält. Ändern Sie also erneut Zeile 40 und 50 in folgende:

```
40 PRINT#7,CHR$(17) CHR$(14) NA$
50 PRINT#7,CHR$(145) CHR$(15):REM GROSSBUCHSTABEN + NORMAL
```

Wenn Sie dieses Programm ablaufen lassen, sehen Sie, daß es möglich ist, mehrere Abweichungen vom Standard gleichzeitig durchführen können. An manchen Druckern, wie z.B. beim EPSON MX-80FT mit GRAFTRAX PLUS, ist es möglich, nicht nur die Breitschrift, sondern auch Schrägschrift, Schmalschrift, Doppelanschlag, Schattenschrift, Hoch- und Tiefstellung und jede Verknüpfung all dieser Funktionen untereinander zu wählen. Mit CHR$ können all die verschiedenen Schriftarten einzeln oder in jeder gewünschten Kombination verwendet werden.

Da wir nun die verschiedenen Wege zur Handhabung der Schriftarten kennen, lassen Sie sie uns praktisch anwenden. Wir schreiben ein Adressen-Druckprogramm für den VIC-1515/1525-Drucker. Manche Hersteller von Aufklebern stellen gummierte Aufkleber auf Lochrand her, so daß Sie diese wie normale Endlosformulare in den Drucker einspannen können. Unser Programm druckt die Namen mit Groß- und Kleinbuchstaben in Breitschrift, die Straße in Standardschrift mit Klein- und Großbuchstaben, und die Postleitzahl und den Wohnort nur in Großbuchstaben.

```
10 PRINT CHR$(147):PRINT CHR$(14):REM GROSS/KLEIN
20 OPEN7,4
30 INPUT "NAME";NA$
40 INPUT "STRASSE";ST$
50 PRINT CHR$(142):REM GROSSBUCHSTABEN
60 INPUT "PLZ";PL$
70 INPUT "ORT";WO$
100 PRINT#7,CHR$(17) CHR$(14) NA$
110 PRINT#7,CHR$(15) CHR$(17) ST$
120 PRINT#7,CHR$(145) PL$+" "+WO$
130 CLOSE7
```

Wenn Sie das Programm ablaufen lassen, sehen Sie, daß Sie auf dem Bildschirm den gleichen Modus wie bei der Druckausgabe haben. Dies weist den Benutzer darauf hin, wie die Druckausgabe erscheint. Beachten Sie, daß für den Groß-/Kleinbuchstabenmodus auf dem Bildschirm unterschiedliche CHR$-Kommandos verwendet werden. So bringt z.B. CHR$(12) den Bildschirm in den Groß-/Kleinschreibmodus, während CHR$(17) den Drucker in diesen Modus bringt.

Um die praktische Anwendung des Programms zu erhöhen, fügen wir noch einige Leerzeilen an das Ende des Drucks, so daß die Ausrichtung auf die Aufkleber genau übereinstimmt. Je nach der Größe Ihrer Aufkleber benötigen Sie weniger oder mehr Leerzeilen. Fügen Sie folgende Zeile an Ihr Programm und gleichen Sie die Schleife an, um die Ausgabe genau auf Ihre Aufkleber auszurichten.

```
125 FOR I = 1 TO 3:PRINT#7:NEXT:REM ÄNDERN SIE "3" IN DIE
KORREKTE ANZAHL VON LEERZEILEN FÜR IHRE AUFKLEBER
```

In Kapitel 8 habe ich versprochen, eine Unterroutine für das "FILE MASTER"-Programm bereitzustellen, um die Namen und Adressen auf dem Drucker auszugeben. Nun, das ist genau das, was wir gerade vorhaben. Um die Änderung durchzuführen, laden Sie Ihr "FILE

MASTER"-Programm in den Speicher und fügen Sie folgende Zeilen hinzu oder ändern Sie sie. Aber tippen Sie nicht alles nochmal ein!

```
470 PRINT "MIT BELIEBIGER EINGABE ZURÜCK ZUM MENÜ"
475 PRINT "MIT 'P' ZUM DRUCKZWEIG"
485 IF A$ = "P" THEN GOSUB 5000
525 INPUT "BILDSCHIRM 'B' ODER DRUCKER 'D'";OP$
570 IF NA$(I) = NA$ AND OP$ = "B" THEN GOSUB 4000
575 IF NA$(I) = NA$ AND OP$ = "D" THEN GOSUB 6000
5000 REM *** UNTERPROGRAMM DRUCKER ***
5010 OPEN7,4
5020 CMD7
5030 FOR I = 1 TO N%.PRINT NA$(I):PRINT ST$(I)
5040 PRINT PLZ$+" "+WO$
5050 NEXT
5060 PRINT#7
5070 CLOSE7
5080 RETURN
6000 *** DRUCKROUTINEN FÜR EINZELNE DATEI ***
6020 OPEN 7,4
6030 CMD7
6040 PRINT NA$(I):PRINT ST$(I):PRINT PL$(I)+" "+WO$(I)
6050 PRINT#7
6060 CLOSE7
6070 RETURN
```

Manchmal möchten Sie nicht, daß die Druckausgabe ganz am linken Rand des Papiers oder des Aufklebers beginnt. Um die Startposition der Textausgabe festzulegen, benutzen Sie CHR$(16) und erreichen damit einen freien linken Rand. Es gibt verschiedene Möglichkeiten, dies zu tun. Die einfachste Art ist die Angabe von CHR$(16), dahinter in Anführungszeichen die Startposition des Textes, dann folgt der auszugebende Text. Sie müssen dabei eine zweistellige Zahl verwenden. Wenn Sie also 5 Stellen vom linken Rand beginnen möchten, müssen Sie "05", nicht nur "5" angeben.

Probieren Sie zum Beispiel folgendes:

```
OPEN7,4
PRINT#7, CHR$(16)"05FÜHRT DIES BERECHNUNGEN DURCH?"
CLOSE7
```

Wie Sie beim Ausführen der Kommandos sehen, gibt Ihr Drucker nur "FÜHRT DIES BERECHNUNGEN DURCH?" aus, und nicht die "05", auch wenn sie mit in den Anführungszeichen aufgeführt ist. "05" stellt nur die Position des Textes ein. Fügen Sie eine weitere "5" ein, so daß die Zeile folgendermaßen aussieht:

```
PRINT#7,CHR$(16)"055FÜHRT DIES BERECHNUNGEN DURCH?"
```

und Sie erhalten "5FÜHRT DIES BERECHNUNGEN DURCH?", da hinter der zweiten Nummer alle Angaben als Informationen interpretiert und somit ausgegeben werden.

Sie können die Anzahl der Leerstellen aber auch CHR$ angeben, anstatt mit Zeichen in Anführungszeichen. Möchten Sie z.B. den gedruckten Text an unterschiedlichen Positionen, festgelegt durch eine Schleife, so ausgeben, daß die nächste Position abhängig von der letzten angesteuert wird, dann legen Sie dies durch CHR$(I) fest, wobei "I" den jeweiligen Schleifenbeginn darstellt. Dies können Sie machen. Es ist jedoch etwas verzwickt, da CHR$ den ASCII-Wert der Nummer beinhalten muß. Zum Beispiel sieht eine "05" so aus:

```
CHR$(0) CHR$(53)
```

Wenn Sie das Beispiel oben anwenden, so sieht das so aus:

```
PRINT#7,  CHR$(16) CHR$(0) CHR$(53) "FÜHRT DIES BERECHNUNGEN
DURCH?"
```

Sie müssen beim Berechnen der Position mit Hilfe einer Schleife beide Stellen der betreffenden Position als Distanz zum Schleifenzähler angeben. Für die meisten Anwendungen ist es daher sehr viel einfacher, die Positionsnummer zusammen mit dem auszugebenden Text in Anführungszeichen zu schreiben.

Bevor wir zur Ausgabe der Grafik übergehen, üben wir noch die Anwendung der Druckposition im Programm. Dies ist sehr hilfreich für Listen, in denen eine Aufteilung in Spalten erforderlich ist. Wir zeigen dies am Beispiel einer Verkaufsliste. In der ersten Spalte steht die Artikelbezeichnung, die zweite Spalte enthält den Einkaufspreis und die dritte Spalte den tatsächlich erzielten Verkaufspreis. Wir benutzen INPUT-Anweisungen, so daß alle Angaben über die Tastatur eingegeben werden können und das Programm bei einem wirklichen Verkauf gestartet wird.

```
10 PRINT CHR$(147)
20 PRINT:PRINT:INPUT "WIE VIELE ARTIKEL";N%
25 DIM AR$(N%),AP(N%),SP(N%)
30 PRINT:FOR I = 1 TO N%
40 PRINT "ARTIKEL #";:INPUT AR$(I)
50 INPUT "EINKAUFSPREIS";AP(I)
60 INPUT "VERKAUFSPREIS";SP(I)
70 PRINT
80 NEXT
100 REM *** DRUCKFORMATIERUNG ***
110 OPEN7,4
120 PRINT#7,"ARTIKEL";CHR$(16)"15EINKAUFSPREIS";
130 PRINT#7,CHR$(16) "35VERKAUFSPREIS"
140 PRINT#7,CHR$(10):REM ZEILENVORSCHUB
150 FOR P = 1 TO N%
160 PRINT#7,AR$(P)
170 PRINT#7,CHR$(16) CHR$(49) CHR$(53) "$";AP(P)
180 PRINT#7,CHR$(16) CHR$(51) CHR$(53) "$";SP(P)
190 NEXT
200 CLOSE7
```

Zwei Dinge müssen in diesem Programm beachtet werden. Sehen Sie sich zuerst die Angabe des CHR$-Codes, zur Festlegung der Druckausgabe in Zeile 170 und 180 an. Die Kombination dieser Codes ist identisch mit der Angabe "15" und "35" in Zeile 120 und 130. Zweitens benutzen wir CHR$(10) zum Zeilenvorschub. Wir könnten hier genausogut nur PRINT#7 angeben, um das gleiche Ergebnis zu erhalten. Sie kommen aber vielleicht einmal in eine Situation, wo Sie in der Mitte der Zeile einen Zeilenvorschub brauchen und dann ist CHR$(10) gut einzusetzen. Um ein wirklich schönes Programm daraus zu machen, errechnen Sie die Summe der Einkaufspreise sowie die Summe der erzielten Verkaufspreise aller Artikel. Ebenso wäre es eine interessante Erweiterung, eine vierte Spalte anzulegen, welche die Differenz zwischen dem Einkaufspreis und dem erzielten Verkaufspreis, also den Bruttogewinn enthält. Sie sollten in der Lage sein, dies allein durchzuführen! (Hinweis: Bauen Sie eine weitere Tabelle auf.)

9.3 Das Ausgeben von Grafik

Da wir jetzt wissen, wie wir Text ausgeben, sehen wir uns die Grafikausgabe an. Die am einfachsten auszugebende Grafik ist die Ihrer Tastatur. Mit dem CURSOR UP-Modus, CHR$(145), können wir die Tastengrafik ausgeben. Probieren Sie z.B. folgendes im Kommandomodus aus:

```
OPEN7,4
PRINT#7,CHR$(145) "<COMMODORE-TASTE-B>"
CLOSE7
```

Sie erhalten ein "Schachbrett"-Muster, genau das linke Zeichen der Taste "B". Da der CURSOR-UP-Modus den Standardwert darstellt, ist es nicht notwendig, zur Ausgabe jedes Grafikzeichens von der Tastatur CHR$(145) einzugeben. Um jedoch ganz sicher zu gehen, sollte es einmal irgendwo im Programm stehen.

Um sich die verschiedenen Grafikzeichen Ihrer Tastatur anzusehen, lassen Sie folgendes Programm ablaufen:

```
10 PRINT CHR$(147)
20 OPEN7,4
30 FOR I = 96 TO 127:REM CHR$ BEREICH SET#1
40 PRINT#7,CHR$(145) CHR$(I)
50 NEXT I
60 FOR J = 161 TO 191:REM CHR$ BEREICH SET#2
70 PRINT#7,CHR$(145) CHR$(J)
80 NEXT J
90 CLOSE7
```

Alle Zeichen Ihrer Tastatur werden nun ausgedruckt. Aber mit Geduld und Ausdauer könnten Sie dies auch über die Tastatur erreichen. CHR$(145) ist ein bißchen überflüssig, denn Sie erhalten das gleiche Ergebnis, wenn Sie es entfernen. Geben Sie jedoch CHR$(17) an, erhalten Sie sehr viele Leerstellen, da dies der "Groß-/Klein"-Modus oder CURSOR-DOWN-Modus ist.

Erzeugen Sie Ihre eigenen Grafikzeichen auf dem Drucker

In Kapitel 7 habe ich gezeigt, wie Sie mit Hilfe des übersetzten Binärcodes Sprites erzeugen können. Wir machen hier das gleiche mit Druckgrafik, was allerdings viel einfacher ist. Wir benutzen zu Anfang nur eine 7 x 7-Matrix anstelle der 24 x 21-Matrix, da sie weniger Berechnungen erfordert. Der VIC-1515-Drucker kann Grafikzeichen in einer 7 x 480-Matrix erstellen. Ich schicke Sie jetzt nicht weg, um Zeichenpapier zu holen, sondern wir erstellen unsere Matrix auf dem Drucker und ich erkläre zwischendurch immer, was passiert.

Zu Beginn verwenden wir CHR$(8), um den Grafikmodus festzulegen. Weiterhin "bilden" wir verkettete CHR$, die dann unser Bild enthalten. Da die Bits die einzelnen Nadeln des Druckkopfes in vertikaler Richtung ansprechen, erzeugen wir vertikale Punkte anstatt horizontaler, wie bei unserer Spritegrafik. Wir benutzen jedoch das gleiche Prinzip wie bei der Spritegrafik. Unsere 7 x 7-Matrix sieht jetzt so aus:

```
  1    ---- ---- ---- ---- ---- ---- ----
  2    ---- ---- ---- ---- ---- ---- ----
  4    ---- ---- ---- ---- ---- ---- ----
  8    ---- ---- ---- ---- ---- ---- ----
 16    ---- ---- ---- ---- ---- ---- ----
 32    ---- ---- ---- ---- ---- ---- ----
 64    ---- ---- ---- ---- ---- ---- ----
 + 128
```

Indem Sie "Punkte" in die Leerstellen eintragen, können Sie jetzt Figuren erstellen. Die vertikale Summe plus 128 wird dann in eine Form übersetzt, die der COMMODORE-64 versteht. Möchten Sie z.B. ein Quadrat zeichnen, müssen Sie sowohl die erste und die letzte Spalte, als auch die erste und letzte Reihe füllen. Beginnend bei der ersten Spalte, wäre der Wert 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128, also gleich 255. Die nächsten 5 Spalten hätten dann jeweils einen Punkt in der ersten und letzten Reihe. Ein Punkt in der ersten Reihe wäre 1, ein Punkt in der letzten Reihe 64, plus 128, ergibt 193. Die letzte Spalte hätte den gleichen Wert wie die erste, 255. Sie müßten deshalb ein CHR$ erzeugen, das folgende Werte hat:

```
255 193 193 193 193 193 255
```

Dazu könnten Sie eine Zeile wie folgt schreiben:

```
BOX$ = CHR$(255) + CHR$(193) + CHR$(193) + CHR$(193) +
CHR$(193) + CHR$(193) + CHR$(255)
```

Das kostet eine Menge Zeit. Stattdessen wäre es viel einfacher, die Werte über DATA-Zeilen einzulesen, wie wir es beim Sprite gemacht haben, und den String auf folgende Weise zu verknüpfen.

```
FOR I = 1 TO 7
READ GRAPHIK
GR$ = GR$ + CHR$(GRAPHIK)
NEXT
DATA 255,193,193,193,193,193,255
```

Lassen Sie uns noch einmal das alles in einem Programm zusammenfassen:

```
10 PRINT CHR$(147)
20 FOR I = 1 TO 7:READ GRAPHIK
30 GR$ = GR$ + CHR$(GRAPHIK)
40 NEXT
50 OPEN7,4
60 PRINT#7,CHR$(8) GR$
70 CLOSE7
100 DATA 255,193,193,193,193,193,255
```

Wenn Sie das Programm ablaufen lassen, wird ein kleines Quadrat auf dem Drucker ausgegeben. Das ist nicht sehr aufregend. Sehen wir uns jetzt an, wie wir dieses Quadrat nutzen können, um eine Matrix zur Erzeugung neuer Zeichen zu erstellen. Das folgende Programm liefert eine 7 x 7-Matrix. Es erfordert nur geringe Änderungen am Programm oben.

```
10 PRINT CHR$(147)
20 FOR I = 1 TO 7:READ GRAPHIK
30 GR$ = GR$ + CHR$(GRAPHIK)
40 NEXT
50 OPEN7,4
52 FOR Y = 1 TO 7
54 FOR X = 1 TO 7
60 PRINT#7,CHR$(8) GR$;
62 NEXT X
64 PRINT#7
```

```
66 NEXT Y
70 CLOES7
100 DATA 255,193,193,193,193,193,255
```

Wenn Sie die 7x7-Matix ausgedruckt haben, sehen Sie, wenn alles funktionierte, daß mehr als nötig ausgegeben wurde. Wir brauchen nur eine einfache Trennung zwischen den Zellen. Abgesehen davon, daß auch ein einzelnes Quadrat hilfreich ist, um alle verschiedenen Formen zu erzeugen, können Sie auch exakt die gewünschte Grafik erzeugen. Wenn Sie Ihren Computer das "herausfinden" lassen, ist alles relativ einfach. Zu Beginn teilen wir die Aufgabe in einfache Gruppen. Sie wissen, daß eine vertikale Linie gleich CHR$(255) ist. Wir nennen sie E$. Sie wissen auch, daß CHR$(193) den Boden und Deckel unserer Box darstellt; wenn wir dies jedoch zur Erstellung einer Matrix verwenden, bekommen wir Doppellinien zwischen den Reihen. Wir beginnen deshalb nur mit einer Linie oben. Das ist einfach, da der Punkt "1" darstellt, wir "128" addieren und damit CHR$(129) erhalten. Für die obere Linie brauchen wir 5 dieser Punkte. Wir können dafür eine FOR/NEXT-Schleife bis 5 anwenden. Schließlich brauchen wir am Ende der Matrix eine Grundlinie. Hier zeichnen wir statt einer einfachen, eine Grundlinie, die aus E$ und CHR$(193) besteht, und die wir als TB$ bezeichnen. Unser Plan ist, erst die sechs Linien der Box mit dem Deckel zu zeichnen, und dann für die siebente Linie eine Reihe mit Deckel und Grundlinie. Es ist wichtig zu beachten, daß wir jetzt eine Grafikform benutzen, die größer ist als unsere 7 x 7-Matrix. Hier unser Programm:

```
10 PRINT CHR$(147)
20 E$ = CHR$(255)
30 FOR I = 1 TO 5:T$ = T$ + CHR$(129):nEXT
40 FOR I = 1 TO 5:TB$ = TB$ + CHR$(193):NEXT
50 OPEN7,4
60 FOR Y = 1 TO 6
70 FOR X = 1 TO 7
80 PRINT#7,CHR$(8) E$ T$
90 NEXT X
100 PRINT#7, E$:REM ENDEKENNZEICHEN ANFÜGEN
110 NEXT Y
120 FOR X = 1 TO 7
130 PRINT#7,CHR$(8) E$ TB$;
140 NEXT
150 PRINT#7,E$
160 CLOSE7
```

Da Sie nun eine bessere Vorstellung von dem haben, was erzeugt werden kann, entwerfen Sie selbst ein paar originelle Druckgrafiken.

Wiederholte Grafikausgabe

Das letzte Element, das wir in Verbindung mit dem Drucker durchgehen, ist die wiederholte Ausgabe der Grafik. Durch die Verwendung von CHR$(26) ist es möglich, jedes Grafikzeichen beliebig oft zu wiederholen. Das Format für die Wiederholung erfordert jedoch einige Sorgfalt. Gehen Sie in folgenden Schritten vor:

1. Eintritt in den Grafikmodus mit CHR$(8)

2. Geben Sie das Wiederholungskommando mit CHR$(26)

3. Geben Sie die Anzahl der Wiederholungen durch ein CHR$-Kommando an. Hinweis: Dies ist anders als bei den Positionskommandos. Sie geben nicht den ASCII Code für die Anzahl der Wiederholungen, sondern die dezimale Zahl. Wenn Sie z.B. möchten, daß eine Grafik 20mal wiederholt wird, benutzen Sie CHR$(20).

4. Geben Sie ein Grafikzeichen ein, auf das gewöhnlich der CHR$-Code für das Semikolon <CHR$(59)> folgt, so daß die Wiederholung in der gleichen Zeile ausgeführt wird.

Machen wir nun ein einfaches Programm, welches einen "Balken" mit unterschiedlicher Länge ausgibt. Es zeigt Ihnen, wie Sie ein Programm starten, das eine Balkengrafik mit verschieden hohen Graphen zur Repräsentation Ihrer Daten ausgibt.

```
10 PRINT CHR$(147):PRINT:PRINT
20 INPUT "LÄNGE DES BALKENS";N
30 RP$ = CHR$(8)+CHR$(26)+CHR$(N):REM GRAPH + WIEDERHOLUNGEN
+ ANZAHL
40 VL$ = CHR$(255)+CHR$(59):REM VERTIKALE GRUNDLINIE PLUS
EINEM STRICHPUNKT
50 OPEN7,4
60 PRINT#7,RP$ VL$
70 CLOSE7
```

Beachten Sie, wie schnell die Balken durch den Einsatz der Wiederholfunktion am Drucker ausgegeben werden. Experimentieren Sie mit den Kommandos und probieren Sie sie zusammen mit anderen Druckkommandos aus, um alles, was Sie sehen möchten, in schwarzweiß zu erzeugen.

9.4 Zusammenfassung

Beim Kauf Ihres Druckers dachten Sie vielleicht, er könnte nur, wie eine normale Schreibmaschine, Text ausgeben. Wie Sie jedoch gesehen haben, war dies nur der Beginn. Neben der Textausgabe kann man verschiedene Schriftarten anwenden, eine bestimmte Position zur Textausgabe festlegen und sogar Grafik ausgeben. Sie können nicht nur die Tastaturgrafik ausdrucken, sondern auch eigene Grafiken erzeugen. Keine Schreibmaschine kann das!

Das Geheimnis bei der Benutzung von Druckern mit Ihrem COMMODORE-64 liegt in der CHR$-Funktion. Einerseits werden CHR$ wie der ASCII Code benutzt, genau wie bei der Bildschirmausgabe. Andererseits werden sie entweder als spezielle Druckerfunktionen oder innerhalb bestimmter Befehlsfolgen benutzt, um Druckausgaben zu erzeugen. Leider ist es nicht möglich, Ihren Drucker anzuweisen, alles auf Papier auszugeben, was sich auf dem Bildschirm befindet. Wie auch immer Sie Ihr Programm rund um die Druckausgabe planen, fast alles, was Sie auf dem Bildschirm ausgeben, kann auch ausgedruckt werden.

10

Programmhilfen und Hinweise

Einführung

Nun, wir sind jetzt bereits beim letzten Kapitel. Ich habe die meisten Kommandos für das BASIC des COMMODORE-64 vorgeführt und viele Tricks für die Handhabung gezeigt. Wenn Sie ernsthaft daran interessiert sind, mehr über Ihren Computer zu erfahren und diesen in seiner vollen Leistungsstärke auszunutzen, gibt es noch eine Menge zu lernen. In der Tat ist das letzte Kapitel dazu bestimmt, Ihnen einige Hinweise zu geben, die über den Umfang dieses Buches hinausgehen.

Zuerst möchten wir Sie in die beste Sache seit der Entdeckung des Siliziums - die COMMODORE-64 User Group - einführen. Es handelt sich dabei um Gruppen mit starkem Interesse, den Computergebrauch zu maximieren. Zweitens möchte ich Sie anregen, sich Computerzeitschriften durchzusehen und mehr über Ihren COMMODORE-64 zu lernen. Drittens untersuchen wir einige Sprachen, die neben BASIC auf Ihrem COMMODORE-64 anwendbar sind. BASIC hat viele Vorteile, aber wie jede Programmiersprache, hat es seine Grenzen und Sie sollten wissen, was sonst noch möglich ist.

Zunächst möchte ich Ihnen noch mehr Programme zeigen. Es sind Listings von Programmen, die Sie vielleicht nützlich, lustig oder beides finden werden. Eines der Programme habe ich ausgewählt, um Ihnen einige Anwendungen aus den letzten neun Titeln zu zeigen und Ihr Wissen darüber noch zu vertiefen. Wir sehen uns auch verschiedene Arten von Programmen an, die Sie kaufen können. Sie sind von professionellen Programmierern geschrieben und vereinfachen Ihre Programme wesentlich, wodurch Sie nicht so stark in Anspruch genommen werden. Später werden wir noch die Hardwareumgebung prüfen, die als Ergänzung Ihres COMMODORE-64 möglich ist.

10.1 Commodore-64 User Groups

Das hilfreichste, nützlichste und wirtschaftlichste nach dem Kauf Ihres COMMODORE-64 ist der Beitritt zu einer COMMODORE-64 User Group. Sie treffen dort nicht nur eine große Anzahl von COMMODORE-64-Anwendern, sondern Sie lernen auch das Programmieren und generell, was Sie mit Ihrem Computer tun oder besser: lassen

sollten. Der Club in Ihrer Umgebung hat vielleicht auch Anwender von anderen Commodore-Computern, z.B. PET oder VIC-20.

Gewöhnlich ist der beste Weg, Kontakt mit einer der COMMODORE-64 User Groups aufzunehmen, der über den örtlichen Computerladen. Sehr oft verkaufen diese Computerläden nicht nur den COMMODORE-64, sondern bieten auch Anwendungen dafür an. Manche organisieren sogar Clubtreffen. Andere Microcomputerclubs haben vielleicht auch COMMODORE-64-Anwender. Wenn sich jedoch in Ihrer Umgebung kein spezieller COMMODORE-Club befindet, schließen Sie sich einer allgemeinen Computer-Gruppe an.

Um Ihre eigene COMMODORE-64 User Group zu gründen, schicken Sie eine Nachricht mit Angabe von Treffpunkt und Zeit an das örtliche Computergeschäft und bitten Sie um Veröffentlichung des Schreibens. "Ihr" Club wird dann in den Zeitschriften aufgeführt und es werden sich weitere Personen finden, die sich Ihrem Club anschließen wollen. Wohnen Sie in einem Gebiet mit nur wenigen COMMODORE-64-Besitzern oder in einer relativ kleinen Stadt, wäre es gut, sich einer größeren Gruppe anzuschließen.

10.2 Commodore-64 Zeitschriften

Es gibt verschiedene vierzehntägig oder monatlich erscheinende Computerzeitschriften, die Informationen über den COMMODORE-64 bieten. Einige dieser Zeitschriften sind allgemein gehalten, andere speziell für den COMMODORE-64. Für Beginner ist es vielleicht eine ganz gute Idee, sich anfangs an die speziellen COMMODORE-64-Hefte zu halten, da jeder Computer seine eigene BASIC-Version hat. Wir geben Ihnen hier ein kurze Übersicht über Hefte, die Ihre Anwendung auf dem COMMODORE-64 unterstützen:

64-er: DAS MAGAZIN FÜR COMPUTER-FANS
Herausgeber:
Redaktion "64-er"
Markt & Technik Verlag Aktiengesellschaft
Hans-Pinsel-Str. 2
8013 Haar bei München

"64-er" erscheint monatlich, jeweils Mitte des Vormonats. Es enthält Tests und Informationen über neu erschienene Hardware.

Weiterhin finden Sie Programmiertechniken, Tips und Tricks für Anfänger, Programmlistings, Spiele und Grafikanwendungen, sowie viele interessante Artikel. Das Einzelheft kostet DM 6,--. Der Abonnementpreis beträgt im Inland DM 72,-- für 12 Ausgaben pro Jahr.

*HAPPY-*COMPUTER
Herausgeber:
Markt & Technik Verlag Aktiengesellschaft
Hans-Pinsel-Str. 2
8013 Haar bei München

"HAPPY-COMPUTER" erscheint monatlich, jeweils Mitte des Vormonats. Es ist allgemein für Homecomputer geschrieben, enthält jedoch viel Information über Commodore-Computer. Inhaltlich ist es dem "64-er" Heft ähnlich. Das Einzelheft kostet DM 5,--. Der Abonnementpreis beträgt im Inland 55,-- DM pro Jahr für 12 Ausgaben.

C H I P Das Computer-Magazin
Herausgeber:
Vogel-Verlag KG Würzburg
Redaktion:
Redaktion CHIP
Bavariaring 8
8000 München 2

"CHIP" erscheint monatlich. Es ist ein allgemeines Magazin für Microcomputer, das unter anderem Informationen und Programmlistings für Commodore-Computer bietet. Das Einzelheft kostet im Inland DM 6,--. Der Preis für das Jahresabonnement für das Inland beträgt DM 66,--.

Es gibt jedoch noch viele andere mehr. Sie finden eine große Auswahl in Computershops und im Zeitschriftenhandel. Dies sollte nur eine Anregung darstellen.

10.3 Der Commodore-64 spricht mehrere Sprachen

Außer in BASIC kann Ihr Computer in verschiedenen anderen Sprachen programmiert werden und diese auch ausführen. In einigen Fällen erfordert dies eine spezielle Hardware und vor allem die entsprechende Software. Wir sehen uns einige dieser Sprachen an.

Eine babylonische Sprachenvielfalt.

ASSEMBLER

Assembler ist eine maschinenorientierte Sprache. Sie ist schneller als BASIC und jede andere Sprache, die wir uns noch ansehen. Um Programme in Assembler zu erstellen, ist es nötig, einen "Assembler" für die Codeeingabe zu besitzen. Diese Sprache ermöglicht Ihnen eine bessere Kontrolle über Ihren COMMODORE-64 als BASIC, ist jedoch schwerer zu erlernen, da ein Assemblerprogramm mehr Instruktionen benötigt als BASIC. (Der Objektcode ist jedoch kompakter und benötigt weniger Sektoren auf Ihrer Diskette.) Der Assembler der Commodore Inc., ASSEMBLER 64, braucht die 1541-Disketteneinheit, da sich der Macroassembler und Dienstprogramme auf der Diskette befinden. Er enthält zwei maschinensprachliche Monitorprogramme, Editor- und Ladeprogramme.

Um die Assemblersprache zu erlernen, werden verschiedene Bücher für den COMMODORE-64 am Markt angeboten. Ferner erhalten Sie auch Hilfe über User Groups. Verschiedene Institutionen bieten dazu auch Lehrgänge an.

Maschinenorientierte und höhere Programmiersprachen

Stellen Sie sich unter den Ausdrücken "Maschinensprache" oder "höhere Programmiersprache" folgendes vor: Unter einer höher entwickelten Programmiersprache versteht man eine stark an die englische Sprache angelehnte Programmierung. Assembler stellt eine Maschinensprache dar. Die weiteren Sprachen, die wir uns ansehen, sind höher entwickelte Programmiersprachen.

PASCAL

PASCAL ist eine höhere Programmiersprache, die eigentlich für den Unterricht in strukturierter Programmierung entwickelt wurde. Sie ist schneller als BASIC, andererseits nicht so schwer zu handhaben wie ASSEMBLER. Es ist möglicherweise die bekannteste höher entwickelte Programmiersprache neben BASIC. Sie werden verschiedene Versionen von Pascal finden. Die Sprache ist jedoch ziemlich gut standardisiert, so daß jede Version von Pascal mit anderen Pascalprogrammen zusammenarbeiten kann.

Auch hierzu sind verschiedene Bücher erhältlich, um das Programmieren in Pascal zu erlernen.

FORTH

FORTH ist eine sehr schnelle höhere Programmiersprache, die entwickelt wurde, um Programme zu erstellen, die fast so schnell wie ASSEMBLER-Programme sind, aber weniger Zeit zur Programmerstel-

lung benötigen. Schneller als PASCAL, BASIC, FORTRAN, COBOL und jede andere höhere Programmiersprache, wird FORTH durch Definieren von "Wörtern" programmiert, welche Routinen ausführen. Vom Anwender selbst definierte Befehle werden mit den bereits implementierten zu FORTH-Programmen verbunden. Das beste an FORTH ist, daß mehrere Implementierungen "offene Versionen" darstellen. Die FIG (FORTH Interset Group) FORTH Version liegt in diesem offenen Bereich. Wenn Sie in Assembler geübt sind, wird es Ihnen auch gelingen, Ihre eigene Version zu installieren.

CP/M

Für den COMMODORE-64 sind verschiedene exzellente CP/M-Programme erhältlich. CP/M hat eine der größten Bibliotheken für alle Sprachen. Um CP/M für Ihren Computer benutzen zu können, brauchen Sie ein Z-80-Microprozessor-Steckmodul, von denen verschiedene erhältlich sind. Viele Geschäftspakete, einschließlich Textverarbeitung und Datenbankprogrammen, sind für CP/M zu bekommen. Für alle, die an professionellen Anwendungen interessiert sind, ist CP/M unbedingt sehenswert. Der COMMODORE-64 hat ein einsteckbares Z-80 Paket, das nur hinten in Ihren Computer gesteckt wird. Es wird mit einer Diskette geliefert, auf der sich das CP/M-Betriebssystem befindet. Wenn Sie CP/M auf Ihrem COMMODORE-64 laufen lassen, übernimmt der Z-80-Microprozessor die Operationen vom 6510. Mit der Commodore Z-80-Erweiterung, haben Sie im Prinzip zwei Computer. Der Commodore Z-80 und das Z-80-Handbuch sind über Ihren Commodore-Händler zu beziehen. Mit der CP/M-Erweiterung können Sie alle unter CP/M lauffähigen Programme installieren und laufen lassen, einschließlich anderen Programmiersprachen, wie PASCAL und FORTH.

Vielseitige Sprachen

Außer den oben beschriebenen Sprachen können Sie Disketten bekommen, die COBOL, FORTRAN, PILOT und andere Sprachen für allgemeine und besondere Anwendungen enthalten. LOGO und PILOT werden zum Beispiel verwendet, um Kinder in die Programmierung einzuweisen, während COBOL hauptsächlich für kommerzielle Anwendungen einge-

setzt wird. Bevor Sie Zeit, Geld und Mühe in eine andere Programmiersprache investieren, empfehlen wir genau zu prüfen, zu welchem Zweck Sie diese brauchen. Wenn Ihr Hauptinteresse in der Entwicklung eigener Programme liegt, lernen Sie zuerst vollständig BASIC und probieren Sie aus, was Sie damit alles machen können. Wenn das voll Ihren Bedürfnissen entspricht und für Ihre Anwendung die Geschwindigkeit eine untergeordnete Rolle spielt, ist es besser, bei BASIC zu bleiben und es auszubauen. Liegt Ihr Hauptinteresse aber in der Verwendung von Anwendungsprogrammen, hängt die Fähigkeit der Sprache von den benutzten Programmen ab. Das wichtigste in diesem Zusammenhang ist die Frage, ob Sie CP/M verwenden oder nicht. Wenn Sie dies möchten, brauchen Sie die Z-80-Erweiterung. Fast alle anderen professionell erstellten Programme laufen auf dem COMMODORE-64 ohne zusätzliche Hardwareerweiterung.

Wenn Sie meinen, daß BASIC die für Sie am besten geeignete Programmiersprache ist, Sie aber Ihre Programme beschleunigen möchten, erreichen Sie dies auf einfachem Weg mit einem Compiler. Im Prinzip ist ein Compiler ein Programm, das Ihren Code in eine binäre Datei umformt, das 4- bis 5mal schneller ist als das COMMODOERE-64 BASIC. Sie müssen dazu nur Programme in BASIC schreiben, diese dann compilieren, und das compilierte Programm abspeichern. Von da an läuft Ihr compiliertes Programm wie ein Programm in Maschinensprache ab.

10.4 Sortier-Routinen

Diese Programme sortieren Strings für Sie. Sie stellen verschiedene Algorithmen dar. Sie können sehen, daß manche Algorithmen schneller arbeiten als andere. Der "Bubble Sort" ist für gänzlich unsortierte Listen der einfachste und langsamste der drei. Er arbeitet aber sehr gut mit teilweise sortierten Listen. "Shell Sort" und "Quick Sort 2" arbeiten generell viel schneller als der "Bubble Sort", benutzen jedoch weit mehr komplexe Formeln. Testen Sie alle drei Typen, um zu sehen, welcher Sort für Ihre Bedürfnisse am besten geeignet ist. Sie brauchen diese Sorts wahrscheinlich in Programmen, die eine alphabetische Reihenfolge für die Verarbeitung voraussetzen.

Bubble Sort

```
10 PRINT CHR$(147)
20 INPUT "WIEVIEL WÖRTER";N%
30 DIM A$(N%+1)
40 FOR N = 1 TO N%
50 INPUT "WORT EINGEBEN";A$(N)
60 Z = Z + 1
70 NEXT N
80 PRINT CHR$(147):FOR I = 1 TO 10 :PRINT:NEXT
83 M$="ALPHABETISCH"
90   L  =  20   - LEN(M$)/2:PRINT   CHR$(18);SPC(L);M$:PRINT
CHR$(146)
100 REM *********************
110 REM BUBBLE SORT
120 REM *********************
130 T = N - 1
140 FLIP = 0:FOR S = 1 TO T:IF A$(S) <= A$(S+1) THEN 160
150 SWITCH$ = A$(S):A$(S) = A$(S+1):A$(S+1)=SWITCH$:FLIP=1:
155 T = S
160 NEXT S:IF FLIP = 1 THEN 140
200 REM *** BILDSCHIRMAUSGABE ***
210 REM *** IN ALPHABETISCHER REIHENFOLGE ***
220 FOR N = 2 TO Z+1
230 F = F + 1
240 IF F > 22 THEN GOSUB 300
250 PRINT A$(N)
260 NEXT N
270 END
300 REM *** PAUSE, WENN BILDSCHIRM VOLL ***
310 PRINT CHR$(18);"MIT BELIEBIGER EINGABE WEITER"
320 GET AN$:IF AN$ = "" THEN 320
330 F = 0:PRINT CHR$(146)
340 RETURN
```

Shell Sort

```
10 PRINT CHR$(147)
20 INPUT "WIEVIEL WÖRTER";N%
30 DIM A$(N%+1)
40 FOR N = 1 TO N%
50 INPUT "WORT EINGEBEN";A$(N)
60 Z = Z + 1
70 NEXT N
80 PRINT CHR$(147):FOR I = 1 TO 10 :PRINT:NEXT
83 M$="ALPHABETISCH"
90   L  =  20   - LEN(M$)/2:PRINT   CHR$(18);SPC(L);M$:PRINT
CHR$(146)
100 REM *********************
110 REM SHELL SORT
120 REM *********************
130 Y = 1
140 Y = 2 * Y:IF Y <= N THEN 40
150 Y = INT(Y / 2).IF Y = 0 THEN 200
160 FOR X = 1 TO N-Y:Z = X
170 K = Y + Z:IF A$(Z) <= A$(K) THEN 190
180 SWITCH$ = A$(Z):A$(Z) = A$(K):A$(K) = SWITCH$
182 Z = Z - Y
184 IF Z > 0 THEN 170
190 NEXT X:GOTO 150
200 REM *** BILDSCHIRMAUSGABE ***
210 REM *** IN ALPHABETISCHER REIHENFOLGE ***
220 FOR N = 2 TO V+1
230 F = F + 1
240 IF F > 22 THEN GOSUB 300
250 PRINT A$(N)
260 NEXT N
270 END
300 REM *** PAUSE, WENN BILDSCHIRM VOLL ***
310 PRINT CHR$(18);"MIT BELIEBIGER EINGABE WEITER"
320 GET AN$:IF AN$ = "" THEN 320
330 F = 0:PRINT CHR$(146)
340 RETURN
```

Quick Sort

```
10 PRINT CHR$(147)
20 INPUT "WIEVIEL WÖRTER";N%
30 DIM A$(N%+1)
40 FOR N = 1 TO N%
50 INPUT "WORT EINGEBEN";A$(N)
60 Z = Z + 1
70 NEXT N
80 PRINT CHR$(147):FOR I = 1 TO 10 :PRINT:NEXT
83 M$="ALPHABETISCH"
90   L  =  20   - LEN(M$)/2:PRINT   CHR$(18);SPC(L);M$:PRINT
CHR$(146)
100 REM **********************
110 REM QUICKSORT 2
120 REM **********************
122 PRINT CHR$(147)
125 S1 = 1
130 L(1) = 1
140 R(1) = N
150 L1 = L(S1)
160 R1 = R(S1)
170 S1 = S1 - 1
180 L2 = L1
190 R2 = R1
200 X$ = A$(INT((L1 + R1)/2))
210 C = C + 1
220 ID A$(L2) >= X$ THEN 250
230 L2 = L2 + 1
240 GOTO 210
250 C = C1
260 IF X$ >= A$(R2) THEN 290
270 R2 = R2 - 1
280 GOTO 250
290 IF L2 > R2 THEN 360
300 S = S + 1
310 T$ = A$(L2)
320 A$(L2) = A$(R2)
330 A$(R2) = T$
340 L2 = L2 + 1
350 R2 = R2 - 1
360 IF L2 <= R2 THEN 210
370 IF L2 >= R1 THEN 410
```

```
380 S1 = S1 + 1
390 L(S1) = L2
400 R(S1) = R1
410 R1 = R2
420 IF L1 < R1 THEN 180
430 IF S1 > 0 THEN 150
440 REM *** SORT BEENDET ***
500 REM *** BILDSCHIRMAUSGABE ***
510 REM *** IN ALPHABETISCHER REIHENFOLGE ***
520 FOR N = 2 TO V+1
530 F = F + 1
540 IF F > 22 THEN GOSUB 300
550 PRINT A$(N)
560 NEXT N
570 END
1000 REM *** PAUSE, WENN BILDSCHIRM VOLL ***
1010 PRINT CHR$(18);"MIT BELIEBIGER EINGABE WEITER"
1020 GET AN$:IF AN$ = "" THEN 320
1030 F = 0:PRINT CHR$(146)
1040 RETURN
```

10.5 Tastaturtricks

Bevor Sie das folgende lesen, versprechen Sie, sich nicht zu ärgern! Einverstanden? Gut, nun können Sie es lesen. Bis zu diesem Punkt haben wir eine Anzahl von Abkürzungen für Ihre Tasten nicht verwendet. Dies war notwendig, damit Sie zuerst in der korrekten Anwendung Ihrer Kommandos Sicherheit bekommen. Sie werden auch sehen, daß Ihnen diese Abkürzungen nicht klar zeigen, was auf Ihrem Computer abläuft, was aber bei der vollen Schreibweise durchaus der Fall ist. Im ANHANG D Ihres COMMODORE-64-HANDBUCHS befindet sich eine Tabelle, die Ihnen zeigt, wie Sie den ersten oder die beiden ersten Buchstaben eines Kommandos, die SHIFT-Taste und dann den zweiten (oder dritten) Buchstaben eingeben müssen, um das gesamte Kommando zu bekommen. Das erspart Ihnen vielleicht etwas Programmierzeit, die Kommandos sind jedoch anfangs ziemlich schwer zu lesen. Bringen Sie zum Beispiel ein Programm in den Speicher, und geben Sie "L<SHIFT-I>(147)" und <RETURN> ein. Dieses Kommando ist das gleiche wie LIST, aber Sie müssen statt vier nur zwei Tasten drücken. Leeren Sie jetzt Ihren Speicher und geben Sie folgendes ein:

```
10 ?C<SHIFT-H>(147):A$ = "IN ORDNUNG"
20 ?S<SHIFT-P>10);R<SHIFT-I>(A$,5)
```

Wissen Sie, bevor Sie RUN eingeben, was passiert? Wenn nicht, machen Sie sich keine Gedanken; es ist auch ein bißchen verwirrend, speziell die Darstellung auf dem Bildschirm. Wenn Sie RUN eingeben, wird der Bildschirm gelöscht und die Nachricht "IN ORDNUNG" am oberen Bildschirmrand, an zehnter Stelle von links, ausgegeben. Listen Sie Ihr Programm auf, und alle Kommandos sind klar. Diese Abkürzungen sind einerseits hilfreich, anderseits unklar. Das LIST-Kommando wird gewöhnlich im Kommandomodus gegeben und es ist geschickt, es in der abgekürzten Form zu verwenden. Ansonsten sind diese Abkürzungen für Anfänger eher verwirrend. Benutzen Sie diejenen, die Ihnen am leichtesten fallen und wenden Sie nach und nach immer mehr Abkürzungen an.

Benutzen von WAIT

Ein Statement, das wir bis jetzt nicht besprochen haben, ist WAIT. Die Beschreibung des Einsatzes von WAIT ist ziemlich kompliziert, und wir werden deshalb nicht so tief einsteigen. Es arbeitet in ähnlicher Weise wie GET; aber es befindet sich kein Cursor auf dem Bildschirm. Es dient einer klaren Benutzerprogrammkontrolle. WAIT kann auch in Spielprogrammen mit dem Joystick dazu benutzt werden, auf eine Eingabe zu warten. Ich zeige Ihnen hier, wie Sie damit ein Programm stoppen, bis eine Taste gedrückt wird. Geben Sie folgendes Programm ein und probieren Sie es aus:

```
10 PRINT CHR$(147)
20 FOR I = 1 TO 10:PRINT:NEXT I
30 PRINT "<DRÜCKEN SIE EINE TASTE UM FORTZUSETZEN>"
40 WAIT 197,64: WAIT 197,64,64
50 PRINT:PRINT "PROGRAMM LÄUFT WEITER ..."
```

Funktionstasten

Rechts auf Ihrer Tastatur befinden sich vier Tasten, die wir bis jetzt nicht beachtet haben. Sie werden "Funktionstasten" genannt und durch CHR$-Werte von 133 bis 140 ausgeführt. Sie werden laufend abgefragt und wenn Sie eine dieser "8" Tasten (4 ohne SHIFT und 4 in Verbindung mit SHIFT) gedrückt haben, verzweigt das Programm in eine Unterroutine. Sie enthalten Anwendungen, die der Benutzer durch den Tastendruck startet, während andere Tasten für die Eingabe von Buchstaben und Grafikzeichen benutzt werden. Nehmen wir an, Sie möchten ein Programm haben, das Namen eingibt, bis eine bestimmte Taste gedrückt wird. Da Sie nicht möchten, daß diese Taste durch eine Eingabetaste für die Namen dargestellt wird, verwenden Sie für diesen Zweck die Funktionstasten. Die CHR$-Werte von 133 bis 140 sind mit den Tasten 1 bis 8 verbunden, die Tasten ohne SHIFT von CHR$(133) bis CHR$(136), die Tasten in Verbindung mit SHIFT von CHR$(137) bis CHR$(140). Zum Beispiel entspricht CHR$(133) der Funktionstaste 1 (f1), CHR$(137) - (f2), CHR$(134) - (f3) usw. und CHR$(140) - (f8). Das folgende Programm zeigt Ihnen, wie Sie ein Programm zur Benutzung der Funktionstasten vorbereiten. Nur die Tasten 1,5 und 8 werden benutzt und das Programm wird beendet, wenn die Taste f5 gedrückt wird.

```
10 PRINT CHR$(147)
20 GET A$
30 PRINT CHR$(19)
40 IF A$ = CHR$(133) THEN GOSUB 1000
50 IF A$ = CHR$(140) THEN GOSUB 2000
60 IF A$ = CHR$(135) THEN END
70 PRINT "WÄHLEN SIE FUNKTIONSTASTE 1 ODER 8"
80 PRINT "BEENDEN MIT FUNKTIONTASTE 5"
90 GOTO 20
1000 PRINT CHR$(147):PRINT "SIE HABEN FT 1 AUSGEWÄHLT"
1010 PRINT "WEITER MIT BELIEBIGER EINGABE"
1020 WAIT 197,64:WAIT 197,64,64
1030 PRINT CHR$(147):RETURN
2000 PRINT CHR$(147):PRINT "SIE HABEN FT 8 AUSGEWÄHLT"
2010 PRINT "WEITER MIT BELIEBIGER EINGABE"
2020 WAIT 197,64:WAIT 197,64,64
2030 PRINT CHR$(147):RETURN
```

10.6 Hilfsprogramme

Was ist ein Hilfsprogramm?

Hilfsprogramme (Utilities) sind Programme, die Sie in der Programmierung oder Ausführung von Systemroutinen unterstützen. Wenn Sie einer COMMODORE-64 User Group angehören, werden Sie mehr über Hilfsprogramme und ihre Anwendung erfahren. Wie bei allen anderen Programmen, die Sie eventuell kaufen möchten, fragen Sie zuerst andere Clubmitglieder und lassen Sie sich diese Programme vorführen!

10.7 Textverarbeitung

Mit einem Textverarbeitungsprogramm kann Ihr COMMODORE-64 in ein erstklassiges Textverarbeitungssystem gewandelt werden. Mit diesem System können Sie alles machen, von der Versetzung von Blökken bis zum Auffinden von bestimmten Wörtern im Text. Sie können

Text in Sekunden editieren und genauso schnell Änderungen vornehmen. Wenn Sie einmal mit Textverarbeitung gearbeitet haben, möchten Sie nie wieder einen Brief mit der Schreibmaschine tippen. Dieses Buch wurde mit Hilfe eines Textverarbeitungsprogramms in einem Bruchteil der Zeit erstellt, die man mit einer Schreibmaschine benötigen würde.

Es gibt auch einige Einschränkungen bei der Anwendung eines Textverarbeitungssystems. Erstens stellt der COMMODORE-64 nur 40 Zeichen am Bildschirm dar. Da der Standardwert 80 Zeichen beträgt, beunruhigt die Darstellung auf dem Papier manche Leute, da diese nicht mit der Darstellung auf dem Bildschirm übereinstimmt. Da ich jedoch Schriften erstelle, die in einer Breite von 40 bis 132 Spalten auf dem Drucker ausgegeben werden, irritiert mich diese Tatsache keineswegs. Wenn Sie 80 Zeichen auf dem Bildschirm dargestellt haben möchten, können Sie einen Adapter kaufen, der Ihnen das ermöglicht. Mit Hilfe des 80-Zeichen-Adapters sehen Sie auf dem Bildschirm exakt das gleiche Bild wie auf Ihrer Druckausgabe. Fragen Sie Ihren örtlichen Commodore-Händler danach.

Im folgenden zeige ich Ihnen einige Möglichkeiten, nach denen Sie bei Ihrem Textverarbeitungsprogramm vielleicht suchen werden:

1. Finden/Ersetzen
   Findet jeden String in Ihrem Text und/oder findet und ersetzt jeden String durch einen anderen. Das ist gut, um Schreibfehler bei einzelnen Wörtern oder ganze Abschnitte auszubessern.

2. Blockbewegungen
   Versetzt Textblöcke von einem Platz an einen anderen. (z.B. versetzt einen Abschnitt aus der Mitte des Textes zum Ende eines Dokumentes.) Eine sehr nützliche Funktion.

3. Einbinden von Dateien
   Verknüpft einzelne Dateien einer Diskette. Dies ist sehr wichtig bei längeren Dokumenten oder für standardisierte Kurzdokumente.

4. Zeilen- und bildschirmorientiertes Editieren
   Das zeilenorientierte Editieren erfordert das Lokalisieren des Zeilenanfangs. Bildschirmorientiertes Editieren erlaubt das Starten des Editiervorgangs an jeder

beliebigen Stelle des Textes. Die zweite Form ist besonders für längere Dokumente von Vorteil.

5. Automatische Seitennumerierung
Seiten werden automatisch durchnumeriert, ohne den Umbruch auf die nächste Seite festlegen zu müssen.

6. Eingebaute Druckersteuerung
Dies ermöglicht dem Benutzer, spezielle Instruktionen direkt an den Drucker zu schicken. Dazu gehören z.B. die "Punktkommandos", die u.a. den linken Rand der Druckausgabe regulieren. Weiterhin gibt es spezielle Kommandos, die die Schriftarten wechseln und vieles andere mehr mit dem auszugebenden Text durchführen, ohne daß man vorher Parameter festsetzen muß. Man kann aber auch gesetzte Parameter überschreiben.

Dies ist nur eine kleine Auswahl der Möglichkeiten, die Sie bei der Textverarbeitung haben. Als Faustregel gilt: je mehr ein Textverarbeitungsprogramm kann, desto mehr kostet es. Wenn Sie nur Briefe und kurze Dokumente schreiben möchten, ist es nicht erforderlich, ein teures Textverarbeitungssystem zu kaufen. Wenn Sie jedoch längere, komplexere und größere Mengen von Dokumenten schreiben, ist die Investition eines höher entwickelten Textverarbeitungsprogramms die Mehrkosten wert. Haben Sie spezielle Anforderungen (z.B. die Erstellung von Rechnungsformularen), sollten Sie nach diesen Möglichkeiten in einem Textverarbeitungsprogramm suchen. Möglicherweise ist ein Programm, das vielleicht bestimmte Bereiche nicht abdeckt, gerade für Ihre speziellen Anforderungen das richtige. Lassen Sie sich, wie bei allen anderen Programmen, eine ausführliche Dokumentation verschiedener Textverarbeitungsprogramme auf dem COMMODORE-64 geben, bevor Sie eines davon kaufen.

Noch ein wichtiger Hinweis: es nimmt einige Zeit in Anspruch, alle Möglichkeiten des Textverarbeitungssystems zu nutzen und es effektiv einzusetzen. Es ist möglich, zu Beginn nur Text einzugeben, während es einiger Praxis bedarf, um alle Leistungsmerkmale voll auszunutzen. Wenn Sie einmal alle Techniken Ihres Textverarbeitungsprogramms kennen, werden Sie schwören, daß Ihr Programm das beste ist! Genießen Sie deshalb mit Vorsicht Argumente über das "optimale Textverarbeitungsprogramm". Es ist wie bei einer Diskussion über Politik oder Religion. Wenn Sie einen Drucker haben, vergewissern Sie sich, daß das Programm auch Text auf Ihren

Drucker ausgibt. Dies ist speziell der Fall, wenn Sie einen parallelen Druckadapter haben.

10.8 Datenbankprogramme

Wenn Sie ein Programm brauchen, das Informationen erzeugt und abspeichert, ist ein "Datenbank"-Programm erforderlich. Grundsätzlich sind professionelle Datenbankprogramme entweder sequentielle Dateien oder mit direktem Zugriff. Wenn Sie das Programm verwenden, benutzen Sie bereits definierte Felder, um gewünschte Felder zu erzeugen oder bereitzustellen. Zum Beispiel hält sich ein Benutzer eine Datenbank für Kunden. Zusätzlich zu den Feldern für Name und Adresse braucht der Anwender vielleicht spezielle Felder für bevorzugte Produkte des Kunden, das Datum der letzten Bestellung, die Höhe des letzten Auftrags, das Datum des letzten Zahlungseingangs, usw.

Datenbankprogramme sollten vor dem Kauf besonders gründlich geprüft werden. Einige der teuren Datenbanken können praktisch mit jeder anderen Anwendung genutzt werden. Wenn Sie jedoch ein Datenbanksystem nur dazu benutzen, eine Liste von Namen und Adressen zu speichern, um Adressenaufkleber zu drucken, macht dies eine speziell dafür entwickelte Datenbank besser, und dazu für weniger Geld. In manchen Fällen leistet das sogar eine Textverarbeitung. Wenn Ihre Bedürfnisse jedoch sehr unterschiedlich sind und die Gestaltung von Masken und variablen Satzlängen erfordern, wählen Sie nicht das einfachste Programm, um diese Arbeit durchzuführen.

10.9 Programme für kommerziellen Einsatz

Kommerzielle Programme gibt es wie Sand am Meer. Es ist das beste, bei der Suche nach dem geeigneten Programm von Ihren speziellen Bedürfnissen auszugehen. Auf der anderen Seite gibt es kommerzielle Programme, die für viele bestimmte Branchen anwendbar sind. Ebenso gibt es verschiedene Kalkulationsprogramme und die oben bereits erwähnten Datenbanken.

Leider geben Geschäftsleute oft große Summen für Systeme aus, die nicht funktionieren. Sie glauben, wenn Soft- und Hardware sehr teuer sind, müssen sie besser als weniger teure Systeme sein. Das kommt wahrscheinlich von der "Sie bekommen nur das, was Sie zahlen"-Mentalität, die zum Kauf von Systemen verleitet, die man gar nicht voll ausnutzen kann.

Da die Computer immer weiter entwickelt und immer preisgünstiger werden, bekommen Sie oft nicht das "für was Sie zahlen", wenn Sie ein sehr teures System kaufen. Oft haben Geschäftsleute am Ende eine riesige Anlage, die überholt, zu groß und zu teuer für ihre Bedürfnisse ist. Einige Computerhändler haben sich auf Geschäftsleute spezialisiert. Sie helfen bei der Auswahl des geeigneten Systems in Ihrer Branche, trainieren das Büropersonal und bieten Wartungsverträge für das System an.

Es ist gut möglich, daß Sie in diesen Spezialgeschäften einen höheren Preis als in einem Diskountladen bezahlen. Sie werden jedoch bei Problemen jeder Art von diesen Spezialgeschäften unterstützt. Da der COMMODORE-64 für Anfänger sehr preisgünstig ist, lohnen sich die Extrakosten, die Sie bei einem Händler zahlen.

Alternativ dazu gibt es auch Unternehmensberater für EDV, die Ihnen bei der Auswahl des geeigneten Systems helfen. Wenn Sie einen Berater suchen, sehen Sie sich nach einem um, der vernünftige Preise hat und unabhängig von einem Hardwarehändler ist. Kontaktieren Sie einen Berater über Telefon, erzählen Sie ihm, daß Sie sich einen COMMODORE-64 kaufen möchten und informieren Sie ihn genau über alle Ihre Anwendungen und Bedürfnisse. Wenn diese mit dem System übereinstimmen, wird er Ihnen die erforderliche Software und die benötigten peripheren Geräte vorschlagen. Wenn er versucht, Ihnen einen anderen Computer vorzuschlagen, ist ihm vielleicht das System unbekannt, und Sie sollten noch einen weiteren Berater befragen.

Wenn Ihnen aber mehrere Berater unabhängig voneinander mitteilen, daß der COMMODORE-64 nicht das richtige System für Ihre Bedürfnisse ist, brauchen Sie tatsächlich eine größere Anlage.

Das soll nicht zynisch klingen, aber ich habe schon viele unglückliche Geschäftsleute getroffen, die das falsche System gekauft haben. Ein Geschäftsmann erzählte mir, daß er eine Anlage für DM 35.000,-- gekauft hat, die seine Anforderungen nicht erfüllen konnte. Schließlich entschied er sich für einen Microcom-

puter, der etwa ein Zehntel des Preises kostet und alles war in bester Ordnung. Damit ist natürlich nicht gesagt, daß man in manchen Branchen nicht eine teurere EDV-Anlage benötigt, um damit bestimmte Funktionen ausführen zu können; denn auch der COMMODORE-64 hat seine Grenzen.

Bevor Sie sich ein System kaufen, vergewissern Sie sich, daß es Ihre Aufgaben übernehmen kann, und überzeugen Sie sich, daß es in einer Weise arbeitet, mit der Sie zufrieden sind. Häufig finden Sie preisgünstige neue Micros, wie den COMMODORE-64, die wirklich besser arbeiten als kostspielige große Maschinen.

10.10 Grafik Pakete

In unserem Kapitel über Grafik habe ich die Grafikmöglichkeiten des COMMODORE-64 demonstriert. Bestimmte Anwendungen erfordern jedoch entweder weiter entwickelte Programmierfähigkeiten oder gute Grafikpakete. So ist es zum Beispiel möglich, mit hochauflösender Grafik zu zeichnen wie mit einer Farbpalette. Die erzeugten Bilder können auf Diskette abgespeichert oder über einen Drucker ausgegeben werden.

10.11 Hardware

Der COMMODORE-64 ist "ausbaufähig". Das bedeutet, daß Sie verschiedene Erweiterungen einbauen und damit seine Leistungsfähigkeit erheblich erhöhen können. An der Rückseite Ihres Gerätes finden Sie drei freie Zugänge, in die Sie Erweiterungseinheiten einstecken können. Weiterhin sind auf der rechten Seite zwei Anschlüsse für Paddles und/oder Joystick, die sowohl für Spielprogramme als auch für alle anderen eingesetzt werden können. Bei Spielprogrammen bewegt man mit ihrer Hilfe Raketen, Raumschiffe und andere Symbole. Sie werden aber auch dazu benutzt, Grafik zu erstellen.

Weitere Hardwareerweiterungen sind Schnittstellen für unterschiedliche periphere Geräte. Eine davon, genannt IEEE-Interface, kann bis zu 15 (!) Geräte an Ihren COMMODORE-64 anschließen.

Ein weiteres wichtiges Peripheral, das Sie sich ansehen sollten, ist die Parallel-Interface-Platine. Mit dieser können Sie Ihren COMMODORE-64 an verschiedene günstige Drucker mit parallelem Interface anschließen.

Vergewissern Sie sich vor dem Kauf einer Schnittstelle oder eines Zusatzgeräts, daß dieses mit Ihrem Computer zusammen arbeitet. Leider werden viele Hardwareerweiterungen mit einer schlechten Dokumentation geliefert, so daß es oft schwer ist, sie ohne fremde Hilfe richtig zum Laufen zu bringen. Auch hier ist eine User Group von unschätzbarem Wert. Fragen Sie andere Mitglieder nach Erfahrungen mit diesen Zusatzgeräten, bevor Sie sich für eines entscheiden.

10.12 Modems und Datenkommunikation

Die reizvollste Sache, die Sie mit Ihrem Computer machen können, ist die Kommunikation mit einem anderen Computer. Sie können nicht nur mit anderen COMMODORE-64 kommunizieren, Sie können dies auch mit anderen Micros tun und sich sogar mit EDV-Großanlagen in Verbindung setzen. Mit einem Modem und der entsprechenden Software sind Sie dazu in der Lage. Das Modem muß eine FTZ-Nummer vom Fernmeldetechnischen Zentralamt haben, damit Sie es betreiben dürfen. Holen Sie sich darüber bei der Deutschen Bundespost Informationen.

10.13 Zusammenfassung

Das Wichtigste aus diesem Kapitel ist, daß wir Ihnen nur einen groben Überblick über die Möglichkeiten für die Erweiterung Ihres COMMODORE-64 aufgezeigt haben. Es gibt da noch sehr viel mehr als das, was man in einem Kapitel darstellen kann, und wenn Sie Ihren COMMODORE-64 erst besser kennen, werden Sie sehen, daß die Auswahl der Software und der peripheren Geräte nur durch die eigene Verwirrung bei diesem Überangebot begrenzt ist. Es gibt noch andere Einheiten für den COMMODORE-64, die wir hier im Rahmen dieses Buches aber keinesfalls aufführen können.

Wenn Sie dieses Buch durchgearbeitet haben, sollten Sie Grundkenntnisse besitzen und die Fähigkeiten Ihres Computers verstehen. Ob Sie ihn nur für eine einzelne Funktion benutzen oder nun ein Computerfreak sind: jedenfalls ist es erforderlich, daß Sie den gesamten Bereich seiner Fähigkeiten verstehen, um Hilfe für Ihre Arbeit, Weiterbildung oder Ihre Spiele zu bekommen. Der Computer ist kein monströses elektronisches Mysterium, sondern ein vielseitiges Hilfsinstrument. Sie verstehen vielleicht im Moment noch nicht die genaue Arbeitsweise, wissen aber auch wahrscheinlich nicht genau, wie der Motor Ihres Autos bis ins kleinste Detail funktioniert. Doch hat Sie dies nie davon abgehalten, Ihr Auto zu benutzen. Denken Sie bei einem Computer vor allem an eine Maschine, die Sie dahin bringt, wohin **Sie** möchten und sehen Sie ihn nicht als Maschine an, der Sie folgen müssen.

Anhang A

C-64 WEDGE

Die "Test-Demo"-Diskette, die mit dem 1541-Diskettenlaufwerk geliefert wird, hat ein ausgezeichnetes Disk-Operating-System, das Ihnen eine Menge Zeit spart, was die ersten zwei Kapitel angeht. Zuerst laden Sie das C-64 WEDGE indem Sie folgendes eingeben:

```
LOAD "C-64 WEDGE",8<RETURN>
RUN
```

Wenn Sie dies gemacht haben, können Sie Ihre Diskette leichter handhaben. Sie können sich jetzt auch das Directory ansehen, ohne das Programm aus dem Speicher zu löschen. Das Folgende ist eine Zusammenstellung der C-64 WEDGE Kommandos, angefertigt von der COMMODORE USERS GROUP:

LOAD	/
LOAD & RUN	↑
LOAD Adresse	%
SAVE	←
SAVE und zurück	←§0:A A=Programmname
ERROR DISPLAY	§
DIRECTORY	>$ or §$
DIRECTORY für 2 Laufwerke	§$Lw Lw=Laufwerknummer
DIRECTORY für Bereich	§$:A A=alphanumerisches Zeichen
DIRECTORY für Bereich und 2 Lw.	§$Lw:A
RENAME Datei oder Programm	§R0:N=A N=Neu A=Alt
FORMAT Diskette	§N0:nm,id nm=Name id=Id Nummer
SCRATCH Datei oder Programm	§S0:Pg Nm Pg Nm=Programmname
SRATCH Bereich	§S0:eB* eB=erster Buchstabe

Das SCRATCH-Bereich-Kommando löscht alle Dateien auf der Diskette, wenn Sie den ersten Buchstaben nicht angeben. So sind bei der Eingabe von §S0:*, alle Programme zerstört!

Copy Datei A nach Datei B	§C0:A=0:N A=alter Name N=neuer Name
Reset	§U0
Diskette initialisieren	§I
Ausschalten von WEDGE	§Q
Einschalten von WEDGE	SYS 52224

Anhang B

COMMODORE-64 Kommandoübersicht

Diese Übersicht wurde in alphabetischer Reihenfolge erfaßt. Die Beispiele zeigen Ihnen, wie Sie die Kommandos mit geeigneter Syntax einsetzen. Wenn ein Kommando auf verschiedene Weise benutzt werden kann, führen wir mehrere Beispiele auf. Einige Beispiele sind im Kommandomodus, andere, mit Angabe der Zeilennummer, im Programmodus, und einige gemischt aufgeführt. Ergebnisse wurden aufgeführt, um klar zu zeigen, was verschiedene Kommandos in den Beispielen erzeugen. Es wurden auch einige spezielle Kommandos, die im Text nicht vorkamen aufgeführt, um die Übersicht zu komplettieren.

ABS() — Ergibt den absoluten Wert einer Zahl oder Variablen.
```
PRINT ABS(-123.45)
Ergebnis: 123.45
```

AND — Logischer Operator (in IF/THEN-Abfragen).
```
IF A$ <> "Y" AND A$ = "COMMODORE-64" THEN 100
```

ASC() — Ergibt den ASCII-Wert des ersten Zeichens in einem String.
```
PRINT ASC("W") oder:
A$ = "COMMODORE-64":PRINT ASC(A$)
```

ATN() — Ergibt den Arcus Tangens der Zahl oder Variablen
```
PRINT ATN(123).
Ergebnis: 1.56266643
```

CHR$() — Ergibt das Zeichen, das dem dezimalen Wert entspricht.
```
PRINT CHR$(65)
Ergebnis: A
```

CLOSE — Schließt Dateien ab.
```
CLOSE#7
```

CLR — Alle Variablen werden auf Null gesetzt.
```
CLR
```

CMD — Sendet Ausgaben zu einer eröffneten Datei, die mit der entsprechenden Dateinummer definiert wurde.
```
OPEN7,4
CMD7
LIST
```

CONT — Fortsetzung eines Programmes nach einer STOP- oder END-Anweisung.
```
CONT
```

COS() — Ergibt den Cosinus einer Variablen oder Zahl.
```
PRINT COS(123)
Ergebnis: .887968907
```

DATA — Zeichenketten oder Zahlenwerte, die mit READ Variablen zugeordnet werden sollen.
```
DATA 2,345,HALLO,"GEHEN"
```

DEF FN() — Vom Benutzer definierbare Funktionen.
```
DEF FN K(X) = X*X
PRINT FN K(4)
Ergebnis: 16
```

DIM — Legt Größe eines Bereiches fest.
```
DIM A$(100)
```

END — Beendet ein laufendes Programm.
```
END
```

EXP() — Ergibt e mit der angegebenen Potenz.
```
PRINT EXP(5)
Ergebnis: 7.69478526E+23
```

FOR — Legt Start und Ende einer Schleife fest.
```
FOR I = 1 TO 100
```

FRE() — Der freie Hauptspeicherplatz wird angezeigt.
```
PRINT FRE(0)
```

GET — Hält die Ausführung eines Programmes an, bis ein Zeichen von der Tastatur empfangen wurde.

```
GET A$:IF A$ = "" THEN 30
```

GET#

Liest ein Zeichen von einer eröffneten Datei.

```
GET#12,R$(I)
```

GOSUB

Springt zu einem Unterprogramm an der angegebenen Zeilennummer.

```
GOSUB 200
```

GOTO

Verzweigt zur angegebenen Zeilennummer.

```
GOTO 200
```

IF/THEN

Fragt eine logische Bedingung ab.

```
IF A$ = "Q" THEN END
```

INPUT

Hält die Programmausführung an, bis eine Zeichenkette oder Zahl eingegeben und mit RETURN abgeschlossen wurde.

```
INPUT "MELDUNG";VAR
```

INPUT#

Liest Daten von einer eröffneten Datei oder Einheit.

```
INPUT#1,R$(I)
```

INT()

Ergibt den ganzzahligen Wert einer Variablen oder Zahl.

```
PRINT INT(123.45)
Ergebnis: 123
```

LEFT$(,)

Trennt den angegebenen linken Teil einer Zeichenkette ab.

```
A$ = "GUTEN TAG"
PRINT LEFT$(A$,3)
Ergebnis: GUT
```

LEN

Ergibt die Länge einer Zeichenkette.

```
A$ = "WIE LANGE BIN ICH?"
PRINT LEN(A$)
Ergebnis: 18
```

LIST

Listet das im Hauptspeicher stehende Programm.

```
LIST
```

LOAD
Lädt Programm von der angegebenen Einheit.
LOAD "$",8 (INHALTSVERZEICHNIS VON DISKETTE)
LOAD "PROGRAMMNAME",1 oder:
LOAD "PROGRAMMNAME"

LOG()
Ergibt den Logarithmus einer angegebenen Variablen oder Zahl.
PRINT LOG(123)
Ergebnis: 4.81218436

MID$(,s,l)
Trennt den angegebenen Teil aus einer Zeichenkette, beginnend bei der Starposition s in der Länge l.
A$ = "WUNDERVOLL"
PRINT MID$(A$,4,3)
Ergebnis: DER

NEW
Löscht das im Hauptspeicher befindliche Programm.
NEW

NEW(DISKETTE)
Formatiert die Diskette, löscht alle Programme. Die Einheit muß vorher eröffnet worden sein.
(New kann mit N abgekürzt werden.)
OPEN 15,8,15
PRINT#15,"NEW0:DISKETTE,22"

NEXT
Markiert das Ende einer FOR/NEXT-Schleife.
FOR I = 1 TO 100
PRINT "WEITER"
NEXT I

NOT
Logische Verneinung in einem IF/THEN-Statement.
IF A NOT B THEN GOTO 100

ON
Für errechnete Sprünge in einem Programm.
ON A GOSUB 1000,2000,3000

OPEN
Eröffnet Datei oder Einheit für die Bearbeitung.
OPEN1,1,1 "NAMENSLISTE"
OPEN7,4 (DRUCKER)

OR
Logisches ODER in einer IF/THEN-Abfrage.
IF A = 10 OR B = 20 THEN GOTO 100

PEEK — Liest den Hauptspeicherinhalt an der angegebenen dezimalen Adresse.

```
PRINT PEEK(768)
IF PEEK(768) = 5 THEN GOTO 200
```

POKE — Speichert einen Wert an der angegebenen Speicheradresse.

```
POKE 768,10
```

Legt den Wert 10 an der Adresse 768 ab.

POS() — Ergibt die augenblickliche Spaltenposition des Cursors.

```
PRINT "DIESE ZEILE";:PRINT POS(0)
```

Ergebnis: DIESE ZEILE 12

PRINT — Gibt Zeichenketten oder Zahlenwerte auf dem Bildschirm oder Drucker aus. (Abk.: ?)

```
PRINT 1;2;3;"GO";F$;A;N%
```

PRINT# — Sendet Ausgaben zu einer eröffneten Datei oder Einheit.

```
PRINT#1,NA$(I) oder:
OPEN7,4 "HALLO COMMODORE-64"
```

READ — Ordnet Datan einer DATA-Anweisung Variablen zu.

```
READ A:READ B$
DARA 5,"BAD"
```

REM — Bemerkungszeile.

```
REM DIES IST EINE BEMERKUNG
```

RESTORE — Setzt Zeiger für die READ-Anweisung auf das erste Element der angegebenen Zeile.

```
FOR I = 1 TO 5:READ A$(I):NEXT
RESTORE
```

RETURN — Rücksprung aus einem Unterprogramm, in das mit GOSUB verzweigt wurde.

```
RETURN
```

RIGHT$(,)

Trennt den spezifizierten rechten Teil aus einer Zeichenkette.

```
A$ = "MARKT UND TECHNIK":PRINT RIGHT$(A$,11)
Ergebnis:
TECHNIK
```

RND()

Ergibt eine Zufallszahl zwischen 0 und 1.

```
PRINT RND(5)
INT(RND(1)*(N)+1)
ZUFALLSZAHLEN ZWISCHEN 1 UND N
```

RUN

Startet das im Hauptspeicher befindliche Programm.

```
RUN
```

SAVE

Speichert ein Programm auf Diskette oder Band.

```
SAVE "GRAPHIK" (BAND)
SAVE "GRAPHIK",1,1 (BAND MIT BANDENDESPEICHER)
SAVE "GRAPHIK",8 (DISKETTE)
```

SIN()

Ergibt den Sinus einer Variablen oder Zahl.

```
PRINT SIN(123)
Ergebnis:
-.459903491
```

SPC()

Druckt die angegebene Anzahl Leerstellen.

```
PRINT SPC(29)"HERE"
```

SQR()

Ergibt die Quadratwurzel einer Variablen/Zahl.

```
PRINT SQR(64)
```

STOP

Hält die Ausführung eines Programmes vorübergehend an. Das Programm kann mit CONT wieder gestartet werden.

```
STOP
```

STR$()

Wandelt eine Zahl in eine Zeichenkette um.

```
T = 123: T$ = STR$(T): TT$ = "$" + T$ + ".00"
```

SYS

Ruft ein Assembler-Unterprogramm auf, das an der angegebenen Hauptspeicheradresse steht.

```
SYS 58692 (BILDSCHIRM LÖSCHEN UND CURSOR OBEN)
FOR I = 1 TO 20:SYS59626:NEXT
(ROLLT BILDSCHIRM 20 ZEILEN AUFWÄRTS)
```

TAB() Setzt einen horizontalen Tabulator in einer PRINT-Anweisung.

```
PRINT TAB(20);"HIER"
```

TAN() Ergibt den Tangens einer Variablen/Zahl.

```
T = 43:V = 55
R = T + V:PRINT TAN(R)
```

VAL() Wandelt Zeichenkette in einen numerischen Wert um.

```
H$ = "123":PRINT VAL(H$)
```

Anhang C: Zeichencodes

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
@		0	£		28	8		56
A	a	1	]		29	9		57
B	b	2	↑		30	:		58
C	c	3	←		31	;		59
D	d	4	SPACE		32	<		60
E	e	5	!		33	=		81
F	f	6	"		34	>		62
G	g	7	#		35	?		63
H	h	8	$		36			64
I	i	9	%		37	♠	A	65
J	j	10	&		38		B	66
K	k	11	'		39		C	67
L	l	12	(		40		D	68
M	m	13	)		41		E	69
N	n	14	*		42		F	70
O	o	15	+		43		G	71
P	p	16	,		44		H	72
Q	q	17	–		45		I	73
R	r	18	.		46		J	74
S	s	19	/		47		K	75
T	t	20	0		48		L	76
U	u	21	1		49		M	77
V	v	22	2		50		N	78
W	w	23	3		51		O	79
X	x	24	4		52		P	80
Y	y	25	5		53	●	Q	81
Z	z	26	6		54		R	82
[		27	7		55	♥	S	83

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
	T	84			99			102
	U	85			100			103
	V	86			101			104
	W	87			105			117
	X	88			106			118
	Y	89			107			119
	Z	90			108			120
		91			109			121
		92			110			122
		93			111			123
		94			112			124
		95			113			125
SPACE		96			114			126
		97			115			127
		98			116			

Index

Abspeichern 45
Arrow- und Pi-Taste 33
ASCII-Code 143, ff.

Bedingungen 95
  AND/OR/NOT 98
  IF/THEN 95
  Vergleichsoperatoren 95
    Stringvergleiche 106
Bildschirm rollen 137,138
Byte 16

CHR$ 143, ff.
CLR/HOME-Taste 32
COMMODORE-Taste 31
Control-Taste 31,32
CP/M 23,244
Cursortaste 32

DATA 77
Dateiverarbeitung 197, ff.
  Band 197, ff.
  CLOSE 198, ff.
  Diskette 207, ff.
  INPUT# 199, ff.
  OPEN 199, ff.
  PRINT# 199, ff.
  sequentielle 207, ff.
Datenbanken 255
Datenrekorder 17
Diskettenlaufwerk 18
Drucker 20,21,217, ff.
  CHR$ 222
  CMD 219
  CLOSE 221
  ESCAPE Sequenzen 223
  Grafikausgabe 230
    frei def. Grafikzeichen 231, ff.
  Korrespondensdrucker 21
  Matrixdrucker 20
  OPEN 219

PRINT# 220,221
Textausgaben 219
Thermodrucker 21

Editor 49, ff.
Erweiterungen 23

FOR/NEXT 80
Geschachtelte Schleifen 82
Schleifenzähler 86
Schrittweite 83, ff.
Funktionstasten 34

Geschachtelte Schleifen 82
GET 76
Grafik
Bildschirmgrafik 164, ff.
bewegte 172, ff.
farbige 167, ff.
Spritegrafik 181, ff.
bewegte 187, ff.
entwerfen 183, ff.
farbige 189, ff.

Hardware 15
HOME 118

INPUT 74
INST/DEL 33

Klammern 57,58
Kommandomodus 40
Korrespondenzdrucker 21

LIST 44

Mathematische Operationen 55, ff.
Matrixdrucker 20
Modem 23,257
Monitor 19,20

PEEK 147, ff.
POKE 147, ff.
PRINT 40

Programme laden 28
Programmiersprachen
  ASSEMBLER 242
  FORTH 243
  PASCAL 243
Programmodus 40

RAM 15
RESTORE-Taste 32
RETURN-Taste 33
ROM 17
RUN/STOP-Taste 32

SAVE 45
Sichern 47
Software 15
Sortieren 245, ff.
  BUBBLE SORT 246
  QUICKSORT 248,249
  SHELLSORT 247
SPC 118
Stringverarbeitung 120
  Aufteilung 123
    LEFT$ 123, ff.
    MID$ 123, ff.
    RIGHT$ 123, ff.
  Berechnungen 121
  formatieren 121
  Umwandeln 127
    Text in Zahlen 127
    Zahlen in Text 128,129
  Verketten 130, ff.
SYS 154, ff.
Systemcheck 24

TAB 118
Tabellen 107
  DIM 109
  mehrdimensionale 110
Tastatur 30
  Funktionstasten 251
  Tricks 249,250
  WAIT 250
Tastaturgrafik 33,34

Textverarbeitung 252, ff.
Thermodrucker 21
Tongenerator 155, ff.
  ATTACH/DECAY 156
  SUSTAIN/RELEASE 156

Überspielen von Band auf Diskette 29
Unterprogramme in Maschinensprache 154
Unterprogrammtechnik 101
  Gliederung 102
  GOSUB - RETURN 101

Variablen 63
Variablennamen 65
Variablentypen 67
  einfache Genauigkeit 67
  Integer 68
  String 69

Verzweigen 92
  Errechnetes GOTO und GOSUB 103, ff.
  GOTO 92, ff.

Z-80 23
Zahlensysteme 149
  dezimal 149
  hexadezimal 149
Zeilen löschen 50

Aus dem M&T-Buchverlag

CP/M und WordStar Anwenderhandbuch
Best.-Nr. MT 310 — DM 29,80*

Software-Auswahl leicht gemacht
Best.-Nr. MT 340 — DM 58,—*

Hardware-Auswahl leicht gemacht
3. überarbeitete und aktualisierte Ausgabe 1984/85
Best.-Nr. MT 350 — DM 58,—*

Personal Computer Lexikon
Best.-Nr. MT 390 — DM 19,80*

Planen und kalkulieren mit VisiCalc
Best.-Nr. MT 450 — DM 32,—*

Basic ohne Probleme: Bd. 1 Unterweisung
Best.-Nr. MT 480 — DM 36,—*

Basic ohne Probleme: Bd. 2 Übungen
Best.-Nr. MT 490 — DM 26,—*

Basic ohne Probleme: Bd. 3 Programmentwicklung und Datenverwaltung
Best.-Nr. MT 500 — DM 44,—*

Basic ohne Probleme: Bd. 4 Allgemeine Dateiverwaltung am praktischen Beispiel
Best.-Nr. MT 514 — DM 53,—*

Basic-Programme für CBM/VC-20-Computer
Best.-Nr. MT 501 — DM 32,—*

Planen und kalkulieren mit Multiplan
Best.-Nr. MT 502 — DM 58,—*

Der IBM-Personal Computer
Best.-Nr. MT 503 — DM 53,—*

Datenkommunikation und Lokale Computer-Netzwerke
Best.-Nr. MT 504 — DM 58,—*

Software richtig eingekauft
Best.-Nr. MT 505 — DM 34,—*

Wörterbuch der Daten- und Telekommunikation
Best.-Nr. MT 506 — DM 38,—*

Multiplan richtig eingesetzt
Best.-Nr. MT 507 — DM 58,—*

Multiplan richtig eingesetzt — Beispiele auf Diskette (5¼", IBM-PC mit MS-DOS 2.0)
Best.-Nr. MT 623 — DM 48,—*

Personal Computer — das intelligente Werkzeug für jedermann
Best.-Nr. MT 508 — DM 53,—*

SuperCalc richtig eingesetzt
Best.-Nr. MT 511 — DM 58,—*

SuperCalc richtig eingesetzt — Beispiele auf Diskette (5¼", IBM-PC mit MS-DOS 2.0)
Best.-Nr. MT 621 — DM 48,—*

Programme und Tips für VC-20
Best.-Nr. MT 513 — DM 38,—*

Das VC-20 Buch
Best.-Nr. MT 516 — DM 49,—*

Das VC-20 Buch — Beispiele auf Kassette
Best.-Nr. MT 581 — DM 19,90*

Das VC-20 Buch — Beispiele auf Diskette
Best.-Nr. MT 582 — DM 29,90*

Ein-Chip-Mikrocomputer-Handbuch
Best.-Nr. MT 517 — DM 58,—*

Die Btx-Fibel
Best.-Nr. MT 519 — DM 29,80*

Das Datenbanksystem dBASE II
Best.-Nr. MT 524 — DM 68,—*

Basic-80 und CP/M
Best.-Nr. MT 525 — DM 48,—*

Einführung in Datenbanksysteme mit dBASE II
Best.-Nr. MT 526 — DM 68,—*

Einführung in Datenbanksysteme mit dBASE II — Beispiele auf Diskette (5¼", IBM-PC mit MS-DOS 2.0)
Best.-Nr. MT 622 — DM 48,—*

dBASE II richtig eingesetzt
Best.-Nr. MT 541 — DM 68,—*

dBASE II richtig eingesetzt — Beispiele auf Diskette (5¼", IBM-PC mit MS-DOS 2.0))
Best.-Nr. MT 544 — DM 48,—*

Einführung in C
Best.-Nr. MT 561 — DM 69,—*

Mit Lotus 1-2-3 zur integrierten Problemlösung
Best.-Nr. MT 582 — DM 68,—*

Mit Lotus 1-2-3 zur integrierten Problemlösung (Beispiele auf Diskette)
Best.-Nr. MT 647 — DM 58,—*

Basic-Dialekte im Vergleich
Best.-Nr. MT 564 — DM 32,—*

Das Commodore 64-Buch, Bd. 1: Leitfaden für Erstanwender — mit Assembler
Best.-Nr. MT 591 — DM 48,—*

Das Commodore 64-Buch, Bd. 1: Beispiele auf Diskette
Best.-Nr. MT 592 — DM 58,—*

Das Commodore 64-Buch, Bd. 2: Basic-Spiele
Best.-Nr. MT 593 — DM 38,—*

* inkl. MwSt. Unverbindliche Preisempfehlung.

Sie erhalten M&T-Bücher in guten Buchhandlungen, Computershops und Fachabteilungen der Kaufhäuser.*

* Sollten Sie diese Programme ausnahmsweise im Handel nicht beziehen können, so bestellen Sie bitte bei:
Markt & Technik Verlag Aktiengesellschaft, Hans-Pinsel-Str. 2, 8013 Haar, Telefon: 089/4813-220

Aus dem M&T-Buchverlag

Das Commodore 64-Buch, Bd. 2: Beispiele auf Diskette
Best.-Nr. MT 594 DM 56,—*

Das Commodore 64-Buch, Bd. 3: Leitfaden für Fortgeschrittene
Best.-Nr. MT 595 DM 36,—*

Das Commodore 64-Buch, Bd. 3: Beispiele auf Diskette
Best.-Nr. MT 596 DM 58,—*

Das Commodore 64-Buch, Bd. 4: Ein Leitfaden für Systemprogrammierer
Best.-Nr. MT 597 DM 38,—*

Das Commodore 64-Buch, Bd. 4: Beispiele auf Diskette
Best.-Nr. MT 598 DM 58,—*

Das Commodore 64-Buch, Bd. 5: Simon's Basic
Best.-Nr. MT 599 DM 36,—*

Das Commodore 64-Buch, Bd. 5: Beispiele auf Diskette
Best.-Nr. MT 600 DM 56,—*

Das Commodore 64-Buch, Bd. 6: Spiele
Best.-Nr. MT 619 DM 38,—*

Das Commodore 64-Buch, Bd. 6: Beispiele auf Diskette
Best.-Nr. MT 620 DM 58,—*

Computerspiele und Wissenswertes — Commodore 64
Best.-Nr. MT 601 DM 29,80*

Computerspiele und Wissenswertes — Commodore 64: Beispiele auf Diskette
Best.-Nr. MT 602 DM 36,—*

Das große Spielebuch Commodore 64
Best.-Nr. MT 603 DM 29,80*

Das große Spielebuch Commodore 64: Beispiele auf Diskette
Best.-Nr. MT 604 DM 38,—*

Software-Schnellkurs: CP/M
Best.-Nr. MT 605 DM 37,—*

Software-Schnellkurs: MailMerge
Best.-Nr. MT 606 DM 37,—*

Software-Schnellkurs: dBASE II
Best.-Nr. MT 607 DM 37,—*

Software-Schnellkurs: SuperCalc
Best.-Nr. MT 608 DM 37,—*

Software-Schnellkurs: WordStar
Best.-Nr. MT 609 DM 37,—*

Software-Schnellkurs: Multiplan
Best.-Nr. MT 610 DM 37,—*

Software-Schnellkurs: Lotus 1-2-3
Best.-Nr. MT 611 DM 48,—*

Software-Schnellkurs: CP/M 86
Best.-Nr. MT 615 DM 37,—*

Software-Schnellkurs: MS-DOS
Best.-Nr. MT 651 DM 37,—*

Mehr als 32 Basic-Programme für den Commodore 64
Best.-Nr. MT 613 DM 49,—*

Mehr als 32 Basic-Programme für den Commodore 64: Beispiele auf Diskette
Best.-Nr. MT 614 DM 48,—*

Mehr als 32 Basic-Programme für den IBM-PC
Best.-Nr. MT 624 DM 68,—*

Mehr als 32 Basic-Programme für den IBM-PC: Beispiele auf Diskette (5¼" mit MS-DOS 2.0)
Best.-Nr. MT 625 DM 58,—*

MS — DOS
Best.-Nr. MT 616 DM 43,—*

Einführung in Forth
Best.-Nr. 635 DM 58,—*

Lotus 1-2-3 richtig eingesetzt
Best.-Nr. MT 637 DM 68,—*

Lotus 1-2-3 richtig eingesetzt Beispiele auf Diskette (5¼", IBM-PC mit MS-DOS 2.0)
Best.-Nr. MT 636 DM 58,—*

Logo — Grafik, Sprache, Mathematik
Best.-Nr. MT 648 DM 42,—*

WordStar für die Praxis
Best.-Nr. MT 642 DM 54,—*

PC-DOS: Das Betriebssystem des IBM-PC
Best.-Nr. MT 643 DM 58,—*

VisiCalc-Arbeitsblätter
Best.-Nr. MT 645 DM 48,—*

Einführung in SuperCalc
Best.-Nr. MT 646 DM 48,—*

Einführung in SuperCalc: Beispiele auf Diskette
Best.-Nr. MT 666 DM 30,—*

Echtzeit-Betriebssysteme für Mikrocomputer
Best.-Nr. MT 653 DM 68,—*

Commodore 64 — Multiplan
Best.-Nr. MT 655 DM 48,—*

Multiplan deutsch
Best.-Nr. MT 656 DM 58,—*

Basic-Programmierhandbuch
Best.-Nr. MT 658 DM 78,—*

Mit SuperCalc 3 zur integrierten Problemlösung
Best.-Nr. MT 859 DM 58,—*

* inkl. MwSt. Unverbindliche Preisempfehlung.

Sie erhalten M&T-Bücher in guten Buchhandlungen, Computershops und Fachabteilungen der Kaufhäuser.*

* Sollten Sie diese Programme ausnahmsweise im Handel nicht beziehen können, so bestellen Sie bitte bei:
Markt & Technik Verlag Aktiengesellschaft, Hans-Pinsel-Str. 2, 8013 Haar, Telefon: 089/4613-220

Einführungskurs: COMMODORE 64

Dieses Buch soll Ihnen helfen, sich mit Ihrem Commodore 64 rundum vertraut zu machen. In den ersten Kapiteln werden Grundkenntnisse über die Hardware vermittelt, damit Sie Ihren Computer ordnungsgemäß aufstellen, anschließen und bedienen können. Dabei werden auch Diskettenlaufwerke, Drucker und Kassettengeräte in Ihrer Funktion beschrieben.

Als Fortgeschrittener können Sie direkt zu den Kapiteln übergehen, die sich mit der Programmiersprache BASIC beschäftigen. Deshalb werden am Anfang der einzelnen Lernabschnitte Orientierungshilfen gegeben, die den Inhalt kurz beschreiben. Zusätzlich wird der gesamte Stoff am Ende jedes Kapitels in Form einer Zusammenfassung wiedergegeben.

Der ausführliche BASIC-Teil umfaßt die gesamten Einsatzgebiete des Commodore-64-BASIC wie

- Grafik
- Musik
- Dateiverwaltung

mit vielen Beispielen, Programmen, Tabellen und wichtigen Hinweisen, die mit der jeweils ausführlich besprochenen BASIC-Anweisung gekoppelt sind.

Das Buch unterstützt durch seinen Aufbau das direkte Arbeiten am Computer, weist auf häufige Fehler und deren Beseitigung oder Vermeidung hin.

Im abschließenden Teil werden Hardware-Erweiterungen und weitere Programmiersprachen mit deren Einsatzmöglichkeiten für den Commodore 64 vorgestellt.

Drei wichtige Sortierroutinen, häufig benötigte Tabellen, sowie eine Zusammenstellung der BASIC-Anweisungen, -Befehle und -Funktionen verhelfen Ihnen zu einem schnellen Überblick und erleichtern das Programmieren mit dem Commodore 64.

Markt&Technik Verlag Aktiengesellschaft
ISBN 3-89090-017-8